BNSP: an R Package for Fitting Bayesian Semiparametric Regression Models and Variable Selection

by Georgios Papageorgiou

Abstract The R package **BNSP** provides a unified framework for semiparametric location-scale regression and stochastic search variable selection. The statistical methodology that the package is built upon utilizes basis function expansions to represent semiparametric covariate effects in the mean and variance functions, and spike-slab priors to perform selection and regularization of the estimated effects. In addition to the main function that performs posterior sampling, the package includes functions for assessing convergence of the sampler, summarizing model fits, visualizing covariate effects and obtaining predictions for new responses or their means given feature/covariate vectors.

Introduction

There are many approaches to non- and semi-parametric modeling. From a Bayesian perspective, Müller and Mitra (2013) provide a review that covers methods for density estimation, modeling of random effects distributions in mixed effects models, clustering, and modeling of unknown functions in regression models.

Our interest is on Bayesian methods for modeling unknown functions in regression models. In particular, we are interested in modeling both the mean and variance functions non-parametrically, as general functions of the covariates. There are multiple reasons why allowing the variance function to be a general function of the covariates may be important (Chan et al., 2006). Firstly, it can result in more realistic prediction intervals than those obtained by assuming constant error variance, or as Müller and Mitra (2013) put it, it can result in more honest representation of uncertainties. Secondly, it allows the practitioner to examine and understand which covariates drive the variance. Thirdly, it results in more efficient estimation of the mean function. Lastly, it produces more accurate standard errors of unknown parameters.

In the R (R Core Team, 2016) package **BNSP** (Papageorgiou, 2018) we implemented Bayesian regression models with Gaussian errors and with mean and log-variance functions that can be modeled as general functions of the covariates. Covariate effects may enter the mean and log-variance functions parametrically or non-parametrically, with the nonparametric effects represented as linear combinations of basis functions. The strategy that we follow in representing unknown functions is to utilize a large number of basis functions. This allows for flexible estimation and for capturing true effects that are locally adaptive. Potential problems associated with large numbers of basis functions, such as over-fitting, are avoided in our implementation, and efficient estimation is achieved, by utilizing spike-slab priors for variable selection. A review of variable selection methods is provided by O'Hara and Sillanpää (2009).

The methods described here belong to the general class of models known as generalized additive models for location, scale, and shape (GAMLSS) (Rigby and Stasinopoulos, 2005; Stasinopoulos and Rigby, 2007) or the Bayesian analogue termed as BAMLSS (Umlauf et al., 2018) and implemented in package **bamlss** (Umlauf et al., 2017). However, due to the nature of the spike-and-slab priors that we have implemented, in addition to flexible modeling of the mean and variance functions, the methods described here can also be utilized for selecting promising subsets of predictor variables in multiple regression models. The implemented methods fall in the general class of methods known as stochastic search variable selection (SSVS). SSVS has received considerable attention in the Bayesian literature and its applications range from investigating factors that affect individual's happiness (George and McCulloch, 1993), to constructing financial indexes (George and McCulloch, 1997), and to gene mapping (O'Hara and Sillanpää, 2009). These methods associate each regression coefficient, either a main effect or the coefficient of a basis function, with a latent binary variable that indicates whether the corresponding covariate is needed in the model or not. Hence, the joint posterior distribution of the vector of these binary variables can identify the models with the higher posterior probability.

R packages that are related to **BNSP** include **spikeSlabGAM** (Scheipl, 2016) that also utilizes SSVS methods (Scheipl, 2011). A major difference between the two packages, however, is that whereas **spikeSlabGAM** utilizes spike-and-slab priors for function selection, **BNSP** utilizes spike-and-slab priors for variable selection. In addition, Bayesian GAMLSS models, also refer to as distributional regression models, can also be fit with R package **brms** using normal priors (Bürkner, 2018). Further, the R package **gamboostLSS** (Hofner et al., 2018) includes frequentist GAMLSS implementation based

on boosting that can handle high-dimensional data (Mayr et al., 2012). Lastly, the R package **mgcv** (Wood, 2018) can also fit generalized additive models with Gaussian errors and integrated smoothness estimation, with implementations that can handle large datasets.

In **BNSP** we have implemented functions for fitting such semi-parametric models, summarizing model fits, visualizing covariate effects and predicting new responses or their means. The main functions are `mvrm`, `mvrm2mcmc`, `print.mvrm`, `summary.mvrm`, `plot.mvrm`, and `predict.mvrm`. A quick description of these functions follows. The first one, `mvrm`, returns samples from the posterior distributions of the model parameters, and it is based on an efficient Markov chain Monte Carlo (MCMC) algorithm in which we integrate out the coefficients in the mean function, generate the variable selection indicators in blocks (Chan et al., 2006), and choose the MCMC tuning parameters adaptively (Roberts and Rosenthal, 2009). In order to minimize random-access memory utilization, posterior samples are not kept in memory, but instead written in files in a directory supplied by the user. The second function, `mvrm2mcmc`, reads-in the samples from the posterior of the model parameters and it creates an object of class `"mcmc"`. This enables users to easily utilize functions from package **coda** (Plummer et al., 2006), including its `plot` and `summary` methods for assessing convergence and for summarizing posterior distributions. Further, functions `print.mvrm` and `summary.mvrm` provide summaries of model fits, including models and priors specified, marginal posterior probabilities of term inclusion in the mean and variance models and models with the highest posterior probabilities. Function `plot.mvrm` creates plots of parametric and nonparametric terms that appear in the mean and variance models. The function can create two-dimensional plots by calling functions from the package **ggplot2** (Wickham, 2009). It can also create static or interactive three-dimensional plots by calling functions from the packages **plot3D** (Soetaert, 2016) and **threejs** (Lewis, 2016). Lastly, function `predict.mvrm` provides predictions either for new responses or their means given feature/covariate vectors.

We next provide a detailed model description followed by illustrations on the usage of the package and the options it provides. Technical details on the implementation of the MCMC algorithm are provided in the Appendix. The paper concludes with a brief summary.

Mean-variance nonparametric regression models

Let $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ denote the vector of responses and let $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\top$ and $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]^\top$ denote design matrices. The models that we consider express the vector of responses utilizing

$$\boldsymbol{Y} = \beta_0 \mathbf{1}_n + \boldsymbol{X}\boldsymbol{\beta}_1 + \boldsymbol{\epsilon},$$

where $\mathbf{1}_n$ is the usual n-dimensional vector of ones, β_0 is an intercept term, $\boldsymbol{\beta}_1$ is a vector of regression coefficients and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)^\top$ is an n-dimensional vector of independent random errors. Each $\epsilon_i, i = 1, \ldots, n$, is assumed to have a normal distribution, $\epsilon_i \sim N(0, \sigma_i^2)$, with variances that are modeled in terms of covariates. Let $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_n^2)^\top$. We model the vector of variances utilizing

$$\log(\boldsymbol{\sigma}^2) = \alpha_0 \mathbf{1}_n + \boldsymbol{Z}\boldsymbol{\alpha}_1,$$

where α_0 is an intercept term and $\boldsymbol{\alpha}_1$ is a vector of regression coefficients. Equivalently, the model for the variances can be expressed as

$$\sigma_i^2 = \sigma^2 \exp(\boldsymbol{z}_i^\top \boldsymbol{\alpha}_1), i = 1, \ldots, n, \tag{1}$$

where $\sigma^2 = \exp(\alpha_0)$.

Let $\boldsymbol{D}(\boldsymbol{\alpha})$ denote an n-dimensional, diagonal matrix with elements $\exp(\boldsymbol{z}_i^\top \boldsymbol{\alpha}_1/2), i = 1, \ldots, n$. Then, the model that we consider may be expressed more economically as

$$\begin{aligned} \boldsymbol{Y} &= \boldsymbol{X}^* \boldsymbol{\beta} + \boldsymbol{\epsilon}, \\ \boldsymbol{\epsilon} &\sim N(\mathbf{0}, \sigma^2 \boldsymbol{D}^2(\boldsymbol{\alpha})), \end{aligned} \tag{2}$$

where $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}_1^\top)^\top$ and $\boldsymbol{X}^* = [\mathbf{1}_n, \boldsymbol{X}]$.

In the following subsections we describe how, within model (2), both parametric and nonparametric effects of explanatory variables on the mean and variance functions can be captured utilizing regression splines and variable selection methods. We begin by considering the special case where there is a single covariate entering the mean model and a single covariate entering the variance model.

Locally adaptive models with a single covariate

Suppose that the observed dataset consists of triplets $(y_i, u_i, w_i), i = 1, \ldots, n$, where explanatory variables u and w enter flexibly the mean and variance models, respectively. To model the nonparametric effects of u and w we consider the following formulations of the mean and variance models

$$\mu_i = \beta_0 + f_\mu(u_i) = \beta_0 + \sum_{j=1}^{q_1} \beta_j \phi_{1j}(u_i) = \beta_0 + \boldsymbol{x}_i^\top \boldsymbol{\beta}_1, \tag{3}$$

$$\log(\sigma_i^2) = \alpha_0 + f_\sigma(w_i) = \alpha_0 + \sum_{j=1}^{q_2} \alpha_j \phi_{2j}(w_i) = \alpha_0 + \boldsymbol{z}_i^\top \boldsymbol{\alpha}_1. \tag{4}$$

In the preceding $\boldsymbol{x}_i = (\phi_{11}(u_i), \ldots, \phi_{1q_1}(u_i))^\top$ and $\boldsymbol{z}_i = (\phi_{21}(w_i), \ldots, \phi_{2q_2}(w_i))^\top$ are vectors of basis functions and $\boldsymbol{\beta}_1 = (\beta_1, \ldots, \beta_{q_1})^\top$ and $\boldsymbol{\alpha}_1 = (\alpha_1, \ldots, \alpha_{q_2})^\top$ are the corresponding coefficients.

In package **BNSP** we implemented two sets of basis functions. Firstly, radial basis functions

$$\begin{aligned} \mathcal{B}_1 = \{ & \phi_1(u) = u, \phi_2(u) = ||u - \xi_1||^2 \log\left(||u - \xi_1||^2\right), \ldots, \\ & \phi_q(u) = ||u - \xi_{q-1}||^2 \log\left(||u - \xi_{q-1}||^2\right) \}, \end{aligned} \tag{5}$$

where $||u||$ denotes the Euclidean norm of u and $\xi_1, \ldots, \xi_{q-1}$ are the knots that within package **BNSP** are chosen as the quantiles of the observed values of explanatory variable u, with $\xi_1 = \min(u_i)$, $\xi_{q-1} = \max(u_i)$ and the remaining knots chosen as equally spaced quantiles between ξ_1 and ξ_{q-1}.

Secondly, we implemented thin plate splines

$$\mathcal{B}_2 = \left\{ \phi_1(u) = u, \phi_2(u) = (u - \xi_1)_+, \ldots, \phi_q(u) = (u - \xi_q)_+ \right\},$$

where $(a)_+ = \max(a, 0)$ and the knots $\xi_1, \ldots, \xi_{q-1}$ are determined as above.

In addition, **BNSP** supports the smooth constructors from package **mgcv** (e.g., the low-rank thin plate splines, cubic regression splines, P-splines, their cyclic versions, and others). Examples on how these smooth terms are used within **BNSP** are provided later in this paper.

Locally adaptive models for the mean and variance functions are obtained utilizing the methodology developed by Chan et al. (2006). Considering the mean function, local adaptivity is achieved by utilizing a large number of basis functions, q_1. Over-fitting, and problems associated with it, is avoided by allowing positive prior probability that the regression coefficients are exactly zero. The latter is achieved by defining binary variables $\gamma_j, j = 1, \ldots, q_1$, that take value $\gamma_j = 1$ if $\beta_j \neq 0$ and $\gamma_j = 0$ if $\beta_j = 0$. Hence, vector $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_{q_1})^\top$ determines which terms enter the mean model. The vector of indicators $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_{q_2})^\top$ for the variance function is defined analogously.

Given vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$, the heteroscedastic, semiparametric model (2) can be written as

$$\begin{aligned} \boldsymbol{Y} &= \boldsymbol{X}_\gamma^* \boldsymbol{\beta}_\gamma + \boldsymbol{\epsilon}, \\ \boldsymbol{\epsilon} &\sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{D}^2(\boldsymbol{\alpha}_\delta)), \end{aligned}$$

where $\boldsymbol{\beta}_\gamma$ consisting of all non-zero elements of $\boldsymbol{\beta}_1$ and $\boldsymbol{X}_\gamma^*$ consists of the corresponding columns of $\boldsymbol{X}^*$. Subvector $\boldsymbol{\alpha}_\delta$ is defined analogously.

We note that, as was suggested by Chan et al. (2006), we work with mean corrected columns in the design matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$, both in this paper and in the **BNSP** implementation. We remove the mean from all columns in the design matrices except those that correspond to categorical variables.

Prior specification for models with a single covariate

Let $\tilde{\boldsymbol{X}} = \boldsymbol{D}(\boldsymbol{\alpha})^{-1} \boldsymbol{X}^*$. The prior for $\boldsymbol{\beta}_\gamma$ is specified as (Zellner, 1986)

$$\boldsymbol{\beta}_\gamma | c_\beta, \sigma^2, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\delta} \sim N(\boldsymbol{0}, c_\beta \sigma^2 (\tilde{\boldsymbol{X}}_\gamma^\top \tilde{\boldsymbol{X}}_\gamma)^{-1}).$$

Further, the prior for $\boldsymbol{\alpha}_\delta$ is specified as

$$\boldsymbol{\alpha}_\delta | c_\alpha, \boldsymbol{\delta} \sim N(\boldsymbol{0}, c_\alpha \boldsymbol{I}).$$

Independent priors are specified for the indicators variables γ_j as $P(\gamma_j = 1 | \pi_\mu) = \pi_\mu, j = 1, \ldots, q_1$,

from which the joint prior is obtained as

$$P(\boldsymbol{\gamma}|\pi_\mu) = \pi_\mu^{N(\gamma)}(1-\pi_\mu)^{q_1-N(\gamma)},$$

where $N(\gamma) = \sum_{j=1}^{q_1}\gamma_j$.

Similarly, for the indicators δ_j we specify independent priors $P(\delta_j = 1|\pi_\sigma) = \pi_\sigma, j = 1,\ldots,q_2$. It follows that the joint prior is

$$P(\boldsymbol{\delta}|\pi_\sigma) = \pi_\sigma^{N(\delta)}(1-\pi_\sigma)^{q_2-N(\delta)},$$

where $N(\delta) = \sum_{j=1}^{q_2}\delta_j$.

We specify inverse Gamma priors for c_β and c_α and Beta priors for π_μ and π_σ

$$\begin{aligned} &c_\beta \sim \text{IG}(a_\beta, b_\beta), c_\alpha \sim \text{IG}(a_\alpha, b_\alpha),\\ &\pi_\mu \sim \text{Beta}(c_\mu, d_\mu), \pi_\sigma \sim \text{Beta}(c_\sigma, d_\sigma). \end{aligned} \tag{6}$$

Lastly, for σ^2 we consider inverse Gamma and half-normal priors

$$\sigma^2 \sim \text{IG}(a_\sigma, b_\sigma) \text{ and } |\sigma| \sim N(0, \phi_\sigma^2). \tag{7}$$

Extension to bivariate covariates

It is straightforward to extend the methodology described earlier to allow fitting of flexible mean and variance surfaces. In fact, the only modification required is in the basis functions and knots. For fitting surfaces, in package **BNSP** we implemented radial basis functions

$$\begin{aligned} \mathcal{B}_3 = \Big\{\phi_1(\boldsymbol{u}) = u_1, \phi_2(\boldsymbol{u}) = u_2, \phi_3(\boldsymbol{u}) = ||\boldsymbol{u}-\boldsymbol{\xi}_1||^2 \log\left(||\boldsymbol{u}-\boldsymbol{\xi}_1||^2\right), \ldots,\\ \phi_q(\boldsymbol{u}) = ||\boldsymbol{u}-\boldsymbol{\xi}_{q-2}||^2 \log\left(||\boldsymbol{u}-\boldsymbol{\xi}_{q-2}||^2\right)\Big\}. \end{aligned}$$

We note that the prior specification presented earlier for fitting flexible functions remains unchained for fitting flexible surfaces. Further, for fitting bivariate or higher order functions, **BNSP** also supports smooth constructors `s`, `te`, and `ti` from **mgcv**.

Extension to additive models

In the presence of multiple covariates, the effects of which may be modeled parametrically or semi-parametrically, the mean model in (3) is extended to the following

$$\mu_i = \beta_0 + \boldsymbol{u}_{ip}^\top\boldsymbol{\beta} + \sum_{k=1}^{K_1} f_{\mu,k}(u_{ik}), i = 1,\ldots,n,$$

where $\boldsymbol{u}_{ip}$ includes the covariates the effects of which are modeled parametrically, $\boldsymbol{\beta}$ denotes the corresponding effects, and $f_{\mu,k}(u_{ik}), k = 1,\ldots,K_1$, are flexible functions of one or more covariates expressed as

$$f_{\mu,k}(u_{ik}) = \sum_{j=1}^{q_{1k}} \beta_{kj}\phi_{1kj}(u_{ik}),$$

where $\phi_{1kj}, j = 1,\ldots,q_{1k}$ are the basis functions used in the kth component, $k = 1,\ldots,K_1$.

Similarly, the variance model (4), in the presence of multiple covariates, is expressed as

$$\log(\sigma_i^2) = \alpha_0 + \boldsymbol{w}_{ip}^\top\boldsymbol{\alpha} + \sum_{k=1}^{K_2} f_{\sigma,k}(w_{ik}), i = 1,\ldots,n,$$

where

$$f_{\sigma,k}(w_{ik}) = \sum_{j=1}^{q_{2k}} \alpha_{kj}\phi_{2kj}(w_{ik}).$$

For additive models, local adaptivity is achieved using a similar strategy, as in the single covariate case. That is, we utilize a potentially large number of knots or basis functions in the flexible components that appear in the mean model, $f_{\mu,k}, k = 1,\ldots,K_1$, and in the variance model, $f_{\sigma,k}, k = 1,\ldots,K_2$. To avoid over-fitting, we allow removal of the unnecessary ones utilizing the usual indicator variables,

$\boldsymbol{\gamma}_k = (\gamma_{k1}, \ldots, \gamma_{kq_{1k}})^\top, k = 1, \ldots, K_1$, and $\boldsymbol{\delta}_k = (\delta_{k1}, \ldots, \delta_{kq_{2k}})^\top, k = 1, \ldots, K_2$. Here, vectors $\boldsymbol{\gamma}_k$ and $\boldsymbol{\delta}_k$ determine which basis functions appear in $f_{\mu,k}$ and $f_{\sigma,k}$ respectively.

The model that we implemented in package **BNSP** specifies independent priors for the indicators variables γ_{kj} as $P(\gamma_{kj} = 1|\pi_{\mu_k}) = \pi_{\mu_k}, j = 1, \ldots, q_{1k}$. From these, the joint prior follows

$$P(\boldsymbol{\gamma}_k|\pi_{\mu_k}) = \pi_{\mu_k}^{N(\gamma_k)}(1 - \pi_{\mu_k})^{q_{1k} - N(\gamma_k)},$$

where $N(\gamma_k) = \sum_{j=1}^{q_{1k}} \gamma_{kj}$.

Similarly, for the indicators δ_{kj} we specify independent priors $P(\delta_{kj} = 1|\pi_{\sigma_k}) = \pi_{\sigma_k}, j = 1, \ldots, q_{2k}$. It follows that the joint prior is

$$P(\boldsymbol{\delta}_k|\pi_{\sigma_k}) = \pi_{\sigma_k}^{N(\delta_k)}(1 - \pi_{\sigma_k})^{q_{2k} - N(\delta_k)},$$

where $N(\delta_k) = \sum_{j=1}^{q_{2k}} \delta_{kj}$.

We specify the following independent priors for the inclusion probabilities.

$$\pi_{\mu_k} \sim \text{Beta}(c_{\mu_k}, d_{\mu_k}), k = 1, \ldots, K_1 \quad \pi_{\sigma_k} \sim \text{Beta}(c_{\sigma_k}, d_{\sigma_k}), k = 1, \ldots, K_2. \tag{8}$$

The rest of the priors are the same as those specified for the single covariate models.

Usage

In this section we provide results on simulation studies and real data analyses. The purpose is twofold: firstly we point out that the package works well and provides the expected results (in simulation studies) and secondly we illustrate the options that the users of **BNSP** have.

Simulation studies

Here we present results from three simulations studies, involving one, two, and multiple covariates. For the majority of these simulation studies, we utilize the same data-generating mechanisms as those presented by Chan et al. (2006).

Single covariate case

We consider two mechanisms that involve a single covariate that appears in both the mean and variance model. Denoting the covariate by u, the data-generating mechanisms are the normal model $Y \sim N\{\mu(u), \sigma^2(u)\}$ with the following mean and standard deviation functions:

1. $\mu(u) = 2u, \sigma(u) = 0.1 + u$,
2. $\mu(u) = \left\{N(u, \mu = 0.2, \sigma^2 = 0.004) + N(u, \mu = 0.6, \sigma^2 = 0.1)\right\}/4$,
   $\sigma(u) = \left\{N(u, \mu = 0.2, \sigma^2 = 0.004) + N(u, \mu = 0.6, \sigma^2 = 0.1)\right\}/6$.

We generate a single dataset of size $n = 500$ from each mechanism, where variable u is obtained from the uniform distribution, $u \sim \text{Unif}(0, 1)$. For instance, for obtaining a dataset from the first mechanism we use

```
> mu <- function(u) {2 * u}
> stdev <- function(u) {0.1 + u}
> set.seed(1)
> n <- 500
> u <- sort(runif(n))
> y <- rnorm(n, mu(u), stdev(u))
> data <- data.frame(y, u)
```

Above we specified the seed value to be one, and we do so in what follows, so that our results are replicable.

To the generated dataset we fit a special case of the model that we presented, where the mean and variance functions in (3) and (4) are specified as

$$\mu = \beta_0 + f_\mu(u) = \beta_0 + \sum_{j=1}^{q_1} \beta_j \phi_{1j}(u) \quad \text{and} \quad \log(\sigma^2) = \alpha_0 + f_\sigma(u) = \alpha_0 + \sum_{j=1}^{q_2} \alpha_j \phi_{2j}(u), \tag{9}$$

with ϕ denoting the radial basis functions presented in (5). Further, we choose $q_1 = q_2 = 21$ basis functions or, equivalently, 20 knots. Hence, we have $\phi_{1j}(u) = \phi_{2j}(u), j = 1, \ldots, 21$, which results in identical design matrices for the mean and variance models. In R, the two models are specified using

```
> model <- y ~ sm(u, k = 20, bs = "rd") | sm(u, k = 20, bs = "rd")
```

The above formula (Zeileis and Croissant, 2010) specifies the response, mean and variance models. Smooth terms are specified utilizing function `sm`, that takes as input the covariate u, the selected number of knots and the selected type of basis functions.

Next we specify the hyper-parameter values for the priors in (6) and (7). The default prior for c_β is inverse Gamma with $a_\beta = 0.5, b_\beta = n/2$ (Liang et al., 2008). For parameter c_α the default prior is a non-informative but proper inverse Gamma with $a_\alpha = b_\alpha = 1.1$. Concerning π_μ and π_σ, the default priors are uniform, obtained by setting $c_\mu = d_\mu = 1$ and $c_\sigma = d_\sigma = 1$. Lastly, the default prior for the error standard deviation is the half-normal with variance $\phi_\sigma^2 = 2$, $|\sigma| \sim N(0, 2)$.

We choose to run the MCMC sampler for 10,000 iterations and discard the first 5,000 as burn-in. Of the remaining 5,000 samples we retain 1 of every 2 samples, resulting in 2,500 posterior samples. Further, as mentioned above, we set the seed of the MCMC sampler equal to one. Obtaining posterior samples is achieved by a function call of the form

```
> DIR <- getwd()
> m1 <- mvrm(formula = model, data = data, sweeps = 10000, burn = 5000,
+       thin = 2, seed = 1, StorageDir = DIR,
+       c.betaPrior = "IG(0.5,0.5*n)", c.alphaPrior = "IG(1.1,1.1)",
+       pi.muPrior = "Beta(1,1)", pi.sigmaPrior = "Beta(1,1)", sigmaPrior = "HN(2)")
```

Samples from the posteriors of the model parameters $\{\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \boldsymbol{\delta}, c_\beta, c_\alpha, \sigma^2\}$ are written in seven separate files which are stored in the directory specified by argument `StorageDir`. If a storage directory is not specified, then function `mvrm` returns an error message, as without these files there will be no output to process. Furthermore, the last two lines of the above function call show the specified priors, which are $c_\beta \sim \text{IG}(0.5, n/2)$, $c_\alpha \sim \text{IG}(1.1, 1.1)$, $\pi_\mu \sim \text{Beta}(1, 1)$, $\pi_\sigma \sim \text{Beta}(1, 1)$ and $|\sigma| \sim N(0, 2)$, respectively. As we mentioned above, these priors are the default ones, and hence the same function call can be achieved without specifying the last two lines. Here we display the priors in order to describe how users can specify their own priors. For parameters c_β and c_α only inverse Gamma priors are available, with parameters that can be specified by the user in the intuitive way. For instance, the prior $c_\beta \sim \text{IG}(1.01, 1.01)$ can be specified in function `mvrm` by using `c.betaPrior = "IG(1.01,1.01)"`. The second parameter of the prior for c_β can be a function of the sample size n (but only symbol n would work here), so for instance `c.betaPrior = "IG(1,0.4*n)"` is another acceptable specification. Further, Beta priors are available for parameters π_μ and π_σ with parameters that can be specified by the user again in the intuitive way. Lastly, two priors are available for the error variance. These are the default half-normal and the inverse Gamma. For instance, `sigmaPrior = "HN(5)"` defines $|\sigma| \sim N(0, 5)$ as the prior while `sigmaPrior = "IG(1.1,1.1)"` defines $\sigma^2 \sim \text{IG}(1.1, 1.1)$ as the prior.

Function `mvrm2mcmc` reads in posterior samples from the files that the call to function `mvrm` generated and creates an object of class `"mcmc"`. Hence, for summarizing posterior distributions and for assessing convergence, functions `summary` and `plot` from package **coda** can be used. As an example, here we read in the samples from the posterior of $\boldsymbol{\beta}$ and summarize the posterior using `summary`. For the sake of economizing space, only the part of the output that describes the posteriors of β_0, β_1, and β_2 is shown below:

```
> beta <- mvrm2mcmc(m1, "beta")

> summary(beta)

Iterations = 5001:9999
Thinning interval = 2
Number of chains = 1
Sample size per chain = 2500

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

                 Mean       SD  Naive SE Time-series SE
(Intercept)  9.534e-01 0.004399 8.799e-05      0.0002534
u            1.864e+00 0.042045 8.409e-04      0.0010356
sm(u).1      3.842e-04 0.016421 3.284e-04      0.0003284
```

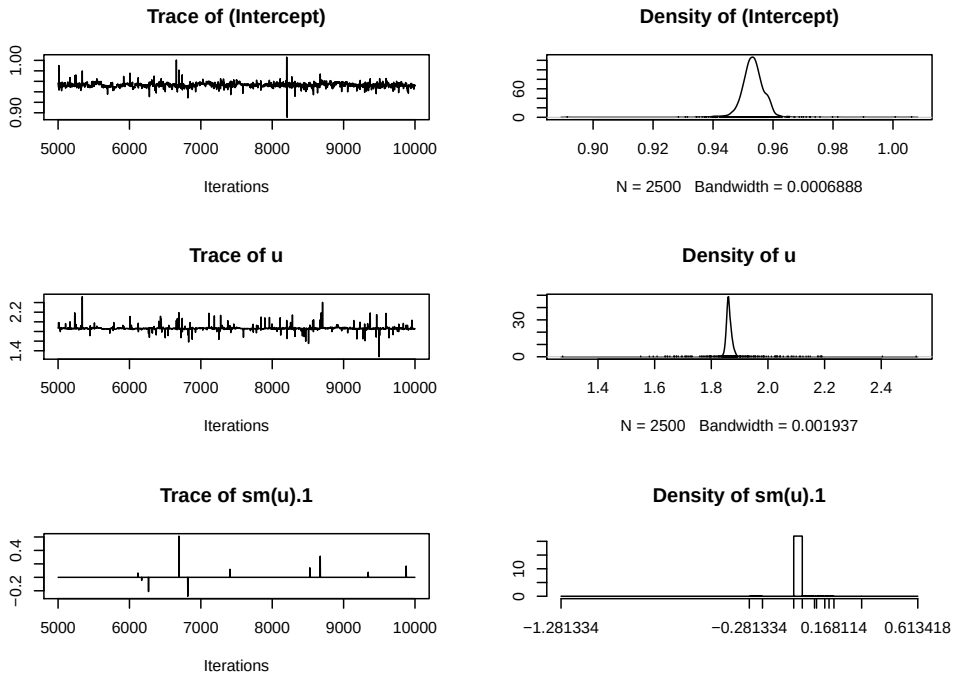


Figure 1: Trace and density plots for the regression coefficients β_0, β_1 and β_2 of the first simulated example. Parameters β_1 and β_2 are the coefficients of the first two basis functions, denoted by "u" and "sm(u).1". Plots for coefficients $\beta_3, \ldots, \beta_{21}$ are omitted as they follow a very similar pattern to that seen for β_2 (i.e., most of the time they take value zero but with random spikes away from zero).

```
2. Quantiles for each variable:

              2.5%    25%    50%    75%  97.5%
(Intercept) 0.946 0.9513 0.9533 0.9554 0.960
u           1.833 1.8565 1.8614 1.8682 1.923
sm(u).1     0.000 0.0000 0.0000 0.0000 0.000
```

Further, we may obtain a plot using

```
> plot(beta)
```

Figure 1 shows the first three of the plots created by function `plot`. These are the plots of the samples from the posteriors of coefficients β_0, β_1 and β_2. As we can see from both the summary and Figure 1, only the first two coefficients have posteriors that are not centered around zero.

Returning to the function `mvrm2mcmc`, it requires two inputs. These are an object of class `"mvrm"` and the name of the file to be read in R. For the parameters in the current model $\{\boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\alpha}, \delta, c_\beta, c_\alpha, \sigma^2\}$ the corresponding file names are 'beta', 'gamma', 'alpha', 'delta', 'cbeta', 'calpha', and 'sigma2' respectively.

Summaries of `mvrm` fits may be obtained utilizing functions `print.mvrm` and `summary.mvrm`. The method `print` takes as input an object of class `"mvrm"`. It returns basic information of the model fit, as shown below:

```
> print(m1)

Call:
mvrm(formula = model, data = data, sweeps = 10000, burn = 5000,
    thin = 2, seed = 1, StorageDir = DIR, c.betaPrior = "IG(0.5,0.5*n)",
    c.alphaPrior = "IG(1.1,1.1)", pi.muPrior = "Beta(1,1)",
    pi.sigmaPrior = "Beta(1,1)", sigmaPrior = "HN(2)")

2500 posterior samples

Mean model - marginal inclusion probabilities
```

```
       u  sm(u).1  sm(u).2  sm(u).3  sm(u).4  sm(u).5  sm(u).6  sm(u).7
  1.0000   0.0040   0.0036   0.0032   0.0084   0.0036   0.0044   0.0028
 sm(u).8  sm(u).9 sm(u).10 sm(u).11 sm(u).12 sm(u).13 sm(u).14 sm(u).15
  0.0060   0.0020   0.0060   0.0036   0.0056   0.0056   0.0036   0.0052
sm(u).16 sm(u).17 sm(u).18 sm(u).19 sm(u).20
  0.0060   0.0044   0.0056   0.0044   0.0052

Variance model - marginal inclusion probabilities
       u  sm(u).1  sm(u).2  sm(u).3  sm(u).4  sm(u).5  sm(u).6  sm(u).7
  1.0000   0.6072   0.5164   0.5808   0.5488   0.6760   0.5320   0.6336
 sm(u).8  sm(u).9 sm(u).10 sm(u).11 sm(u).12 sm(u).13 sm(u).14 sm(u).15
  0.6936   0.6708   0.5996   0.4816   0.4912   0.3728   0.6268   0.5688
sm(u).16 sm(u).17 sm(u).18 sm(u).19 sm(u).20
  0.5872   0.6528   0.4428   0.6900   0.5356
```

The function returns a matched call, the number of posterior samples obtained, and marginal inclusion probabilities of the terms in the mean and variance models.

Whereas the output of the `print` method focuses on marginal inclusion probabilities, the output of the `summary` method focuses on the most frequently visited models. It takes as input an object of class `"mvrm"` and the number of (most frequently visited) models to be displayed, which by default is set to `nModels = 5`. Here to economize space we set `nModels = 2`. The information returned by the `summary` method is shown below

```
> summary(m1, nModels = 2)

Specified model for the mean and variance:
y ~ sm(u, k = 20, bs = "rd") | sm(u, k = 20, bs = "rd")

Specified priors:
[1] c.beta = IG(0.5,0.5*n) c.alpha = IG(1.1,1.1)  pi.mu = Beta(1,1)
[4] pi.sigma = Beta(1,1)   sigma = HN(2)

Total posterior samples: 2500 ; burn-in: 5000 ; thinning: 2

Files stored in /home/papgeo/1/

Null deviance:           1299.292
Mean posterior deviance:  -88.691

Joint mean/variance model posterior probabilities:
  mean.u mean.sm.u..1 mean.sm.u..2 mean.sm.u..3 mean.sm.u..4 mean.sm.u..5
1      1            0            0            0            0            0
2      1            0            0            0            0            0
  mean.sm.u..6 mean.sm.u..7 mean.sm.u..8 mean.sm.u..9 mean.sm.u..10
1            0            0            0            0             0
2            0            0            0            0             0
  mean.sm.u..11 mean.sm.u..12 mean.sm.u..13 mean.sm.u..14 mean.sm.u..15
1             0             0             0             0             0
2             0             0             0             0             0
  mean.sm.u..16 mean.sm.u..17 mean.sm.u..18 mean.sm.u..19 mean.sm.u..20 var.u
1             0             0             0             0             0     1
2             0             0             0             0             0     1
  var.sm.u..1 var.sm.u..2 var.sm.u..3 var.sm.u..4 var.sm.u..5 var.sm.u..6
1           1           1           1           1           1           1
2           1           0           1           1           1           1
  var.sm.u..7 var.sm.u..8 var.sm.u..9 var.sm.u..10 var.sm.u..11 var.sm.u..12
1           1           1           1            0            1            0
2           1           1           1            1            1            1
  var.sm.u..13 var.sm.u..14 var.sm.u..15 var.sm.u..16 var.sm.u..17 var.sm.u..18
1            1            1            1            1            1            1
2            0            1            1            1            1            0
  var.sm.u..19 var.sm.u..20 freq prob cumulative
1            1            1  141 5.64       5.64
2            1            0  120 4.80      10.44
Displaying 2 models of the 916 visited
```

```
2 models account for 10.44% of the posterior mass
```

Firstly, the method provides the specified mean and variance models and the specified priors. This is followed by information about the MCMC chain and the directory where files have been stored. In addition, the function provides the null and the mean posterior deviance. Finally, the function provides the specification of the joint mean/variance models that were visited most often during MCMC sampling. This specification is in terms of a vector of indicators, each consisting of zeros and ones, that show which terms are in the mean and variance model. To make clear which terms pertain to the mean and which to the variance function, we have preceded the names of the model terms by "`mean.`" or "`var.`". In the above output, we see that the most visited model specifies a linear mean model (only the linear term is included in the model) while the variance model includes twelve terms. See also Figure 2.

We next describe the function `plot.mvrm` which creates plots of terms in the mean and variance functions. Two calls to the `plot` method can be seen in the code below. Argument `x` expects an object of class `"mvrm"`, as created by a call to the function `mvrm`. The `model` argument may take on one of three possible values: `"mean"`, `"stdev"`, or `"both"`, specifying the model to be visualized. Further, the `term` argument determines the term to be plotted. In the current example there is only one term in each of the two models which leaves us with only one choice, `term = "sm(u)"`. Equivalently, `term` may be specified as an integer, `term = 1`. If `term` is left unspecified, then, by default, the first term in the model is plotted. For creating two-dimensional plots, as in the current example, the `plot` method utilizes the package **ggplot2**. Users of **BNSP** may add their own options to plots via the argument `plotOptions`. The code below serves as an example.

```
> x1 <- seq(0, 1, length.out = 30)
> plotOptionsM <- list(geom_line(aes_string(x = x1, y = mu(x1)), col = 2, alpha = 0.5,
+                      lty = 2), geom_point(data = data, aes(x = u, y = y)))
> plot(x = m1, model = "mean", term = "sm(u)", plotOptions = plotOptionsM,
+      intercept = TRUE, quantiles = c(0.005, 0.995), grid = 30)
> plotOptionsV = list(geom_line(aes_string(x = x1, y = stdev(x1)), col = 2,
+                     alpha = 0.5, lty = 2))
> plot(x = m1, model = "stdev", term = "sm(u)", plotOptions = plotOptionsV,
+      intercept = TRUE, quantiles = c(0.05, 0.95), grid = 30)
```

The resulting plots can be seen in Figure 2, panels (a) and (b). Panel (a) displays the simulated dataset, showing the expected increase in both the mean and variance with increasing values of the covariate u. Further, we see the posterior mean of $\mu(u) = \beta_0 + f_\mu(u) = \beta_0 + \sum_{j=1}^{21} \beta_j \phi_{1j}(u)$ evaluated over a grid of 30 values of u, as specified by the (default) `grid = 30` option for `plot`. For each sample $\boldsymbol{\beta}^{(s)}, s = 1, \ldots, S$, from the posterior of $\boldsymbol{\beta}$, and for each value of u over the grid of 30 values, $u_j, j = 1, \ldots, 30$, the `plot` method computes $\mu(u_j)^{(s)} = \beta_0^{(s)} + \sum_{j=1}^{21} \beta_j^{(s)} \phi_{1j}(u_j)$. The default option `intercept = TRUE` specifies that the intercept β_0 is included in the computation, but it may be removed by setting `intercept = FALSE`. The posterior means are computed by the usual $\bar{\mu}(u_j) = S^{-1} \sum_s \mu(u_j)^{(s)}$ and are plotted with solid (blue-color) line. By default, the function displays 80% point-wise credible intervals (CI). In Figure 2, panel (a) we have plotted 99% CIs, as specified by option `quantiles = c(0.005,0.995)`. This option specifies that for each value $u_j, j = 1, \ldots, 30$, on the grid, 99% CIs for $\mu(u_j)$ are computed by the 0.5% and 99.5% quantiles of the samples $\mu(u_j)^{(s)}, s = 1, \ldots, S$. Plots without credible intervals may be obtained by setting `quantiles = NULL`.

Figure 2, panel (b) displays the posterior mean of the standard deviation function $\sigma(u) = \sigma \exp\{\sum_{j=1}^{q_2} \alpha_j \phi_{2j}(u)/2\}$. The details are the same as for the plot of the mean function, so here we briefly mention a difference: option `intercept = TRUE` specifies that σ is included in the calculation. It may be removed by setting `intercept = FALSE`, which will result in plots of $\sigma(u)^* = \exp\{\sum_{j=1}^{q_2} \alpha_j \phi_{2j}(u)/2\}$.

We use the second simulated dataset to show how the `s` constructor from the package **mgcv** may be used. In our example, we use `s` to specify the model:

```
> model <- y ~ s(u, k = 15, bs = "ps", absorb.cons=TRUE) |
+              s(u, k = 15, bs = "ps", absorb.cons=TRUE)
```

Function `BNSP::s` calls in turn `mgcv::s` and `mgcv::smoothCon`. All options of the last two functions may be passed to `BNSP::s`. In the example above we used options `k`, `bs`, and `absorb.cons`.

The remaining R code for the second simulated example is precisely the same as the one for the first example, and hence omitted. Results are shown in Figure 2, panels (c) and (d).

We conclude the current section by describing the function `predict.mvrm`. The function provides predictions and posterior credible or prediction intervals for given feature vectors. The two types of intervals differ in the associated level of uncertainty: prediction intervals attempt to capture a future response and are usually much wider than credible intervals that attempt to capture a mean response.

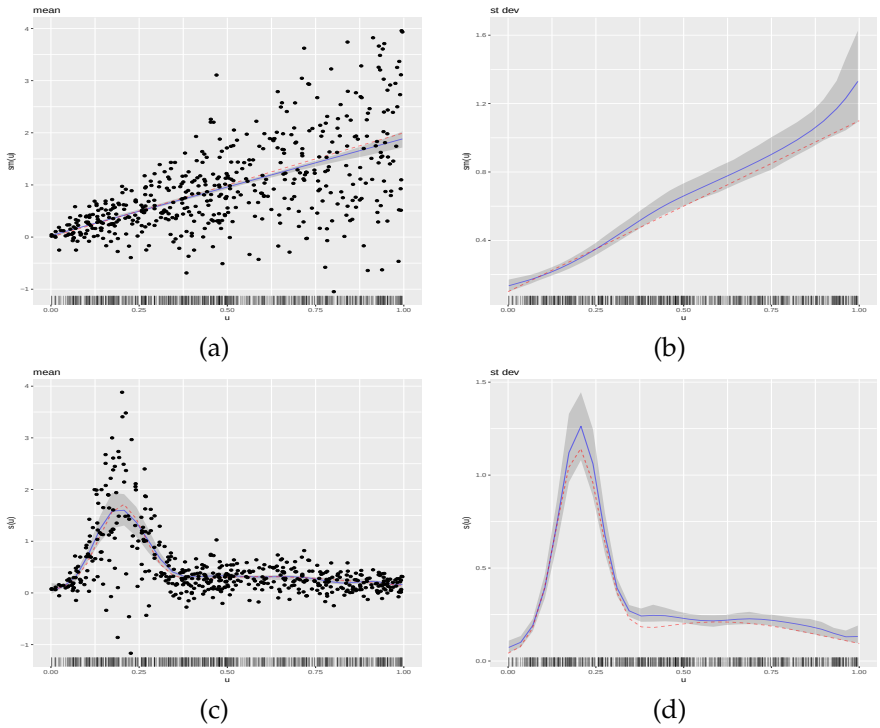


Figure 2: Results from the single covariate simulated examples. The column on the left-hand side displays the generated data points and posterior means of the estimated effect along with 99% CIs. The column on the right-hand side displays the posterior mean of the estimated standard deviation function along with 90% CIs. In all panels, the truth is represented by dashed (red color) lines, the posterior means by solid (blue color) lines, and the CIs by gray color.

The following code shows how credible and prediction intervals can be obtained for a sequence of covariate values stored in `x1`

```
> x1 <- seq(0, 1, length.out = 30)
> p1 <- predict(m1, newdata = data.frame(u = x1), interval = "credible")
> p2 <- predict(m1, newdata = data.frame(u = x1), interval = "prediction")
```

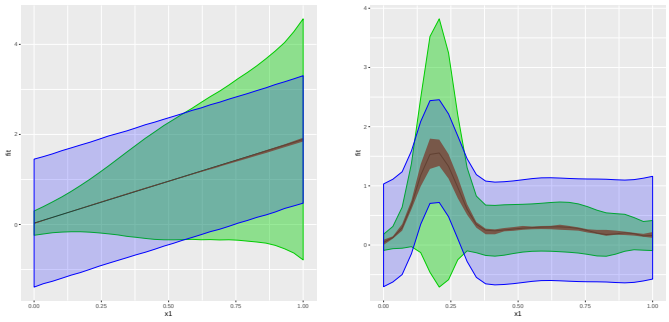
where the first argument in the `predict` method is a fitted `mvrm` model, the second one is a data frame containing the feature vectors at which predictions are to be obtained and the last one defines the type of interval to be created. We applied the `predict` method to the two simulated datasets. To each of those datasets we fitted two models: the first one is the one we saw earlier, where both the mean and variance are modeled in terms of covariates, while the second one ignores the dependence of the variance on the covariate. The latter model is specified in R using

```
> model <- y ~ sm(u, k = 20, bs = "rd") | 1
```

Results are displayed in Figure 3. Each of the two figures displays a credible interval and two prediction intervals. The figure emphasizes a point that was discussed in the introductory section, that modeling the variance in terms of covariates can result in more realistic prediction intervals. The same point was recently discussed by Mayr et al. (2012).

Bivariate covariate case

Interactions between two predictors can be modeled by appropriate specification of either the built-in `sm` function or the smooth constructors from **mgcv**. Function `sm` can take up to two covariates, both of which may be continuous or one continuous and one discrete. Next we consider an example that involves two continuous covariates. An example involving a continuous and a discrete covariate is shown later on, in the second application to a real dataset.



(a) (b)

Figure 3: Predictions results from the first two simulated datasets. Each panel displays a credible interval and two prediction intervals, one obtained using a model that recognizes the dependence of the variance on the covariate and one that ignores it.

Let $\boldsymbol{u} = (u_1, u_2)^\top$ denote a bivariate predictor. The data-generating mechanism that we consider is

$$\begin{aligned}
&y(\boldsymbol{u}) \sim N\{\mu(\boldsymbol{u}), \sigma^2(\boldsymbol{u})\},\\
&\mu(\boldsymbol{u}) = 0.1 + N\left(\boldsymbol{u}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) + N\left(\boldsymbol{u}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right),\\
&\sigma^2(\boldsymbol{u}) = 0.1 + \left\{N\left(\boldsymbol{u}, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1\right) + N\left(\boldsymbol{u}, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2\right)\right\}/2,\\
&\boldsymbol{\mu}_1 = \begin{pmatrix} 0.25 \\ 0.75 \end{pmatrix}, \boldsymbol{\Sigma}_1 = \begin{pmatrix} 0.03 & 0.01 \\ 0.01 & 0.03 \end{pmatrix}, \boldsymbol{\mu}_2 = \begin{pmatrix} 0.65 \\ 0.35 \end{pmatrix}, \boldsymbol{\Sigma}_2 = \begin{pmatrix} 0.09 & 0.01 \\ 0.01 & 0.09 \end{pmatrix}.
\end{aligned}$$

As before, u_1 and u_2 are obtained independently from uniform distributions on the unit interval. Further, the sample size is set to $n = 500$.

In R, we simulate data from the above mechanism using

```
> mu1 <- matrix(c(0.25, 0.75))
> sigma1 <- matrix(c(0.03, 0.01, 0.01, 0.03), 2, 2)
> mu2 <- matrix(c(0.65, 0.35))
> sigma2 <- matrix(c(0.09, 0.01, 0.01, 0.09), 2, 2)
> mu <- function(x1, x2) {x <- cbind(x1, x2);
+       0.1 + dmvnorm(x, mu1, sigma1) + dmvnorm(x, mu2, sigma2)}
> Sigma <- function(x1, x2) {x <- cbind(x1, x2);
+          0.1 + (dmvnorm(x, mu1, sigma1) + dmvnorm(x, mu2, sigma2)) / 2}
> set.seed(1)
> n <- 500
> w1 <- runif(n)
> w2 <- runif(n)
> y <- vector()
> for (i in 1:n) y[i] <- rnorm(1, mean = mu(w1[i], w2[i]),
+                              sd = sqrt(Sigma(w1[i], w2[i])))
> data <- data.frame(y, w1, w2)
```

We fit a model with mean and variance functions specified as

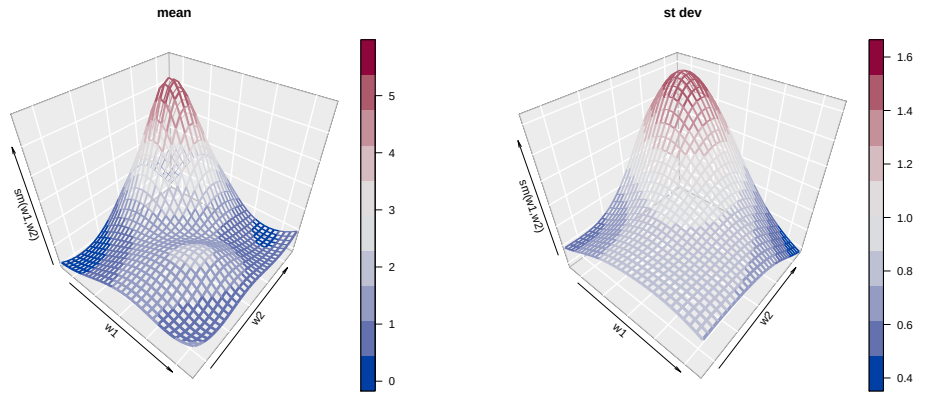
$$\mu(\boldsymbol{u}) = \beta_0 + \sum_{j_1=1}^{12}\sum_{j_2=1}^{12} \beta_{j_1,j_2}\phi_{1j_1,j_2}(\boldsymbol{u}), \quad \log(\sigma^2(\boldsymbol{u})) = \alpha_0 + \sum_{j_1=1}^{12}\sum_{j_2=1}^{12} \alpha_{j_1,j_2}\phi_{2j_1,j_2}(\boldsymbol{u}).$$

The R code that fits the above model is

```
> Model <- y ~ sm(w1, w2, k = 10, bs = "rd") | sm(w1, w2, k = 10, bs = "rd")
> m2 <- mvrm(formula = Model, data = data, sweeps = 10000, burn = 5000, thin = 2,
+            seed = 1, StorageDir = DIR)
```

As in the univariate case, convergence assessment and univariate posterior summaries may be obtained by using function `mvrm2mcmc` in conjunction with functions `plot.mcmc` and `summary.mcmc`. Further, summaries of the `mvrm` fits may be obtained using functions `print.mvrm` and `summary.mvrm`. Plots of the bivariate effects may be obtained using function `plot.mvrm`. This is shown below, where argument `plotOptions` utilizes the package **colorspace** (Zeileis et al., 2009).

```
> plot(x = m2, model = "mean", term = "sm(w1,w2)", static = TRUE,
```



(a) (b)

Figure 4: Bivariate simulation study results with two continuous covariates. Shown are posterior means of (a) the mean and (b) the standard deviation function.

```
+      plotOptions = list(col = diverge_hcl(n = 10)))
> plot(x = m2, model = "stdev", term = "sm(w1,w2)", static = TRUE,
+      plotOptions = list(col = diverge_hcl(n = 10)))
```

Results are shown in Figure 4. For bivariate predictors, function `plot.mvrm` calls function `ribbon3D` from the package **plot3D**. Dynamic plots, viewable in a browser, can be created by replacing the default '`static=TRUE`' by '`static=FALSE`'. When the latter option is specified, function `plot.mvrm` calls the function `scatterplot3js` from the package **threejs**. Users may pass their own options to `plot.mvrm` via the `plotOptions` argument.

Multiple covariate case

We consider fitting general additive models for the mean and variance functions in a simulated example with four independent continuous covariates. In this scenario, we set $n = 1000$. Further the covariates $\boldsymbol{w} = (w_1, w_2, w_3, w_4)^\top$ are simulated independently from a uniform distribution on the unit interval. The data-generating mechanism that we consider is

$$Y(\boldsymbol{w}) \sim N(\mu(\boldsymbol{w}), \sigma^2(\boldsymbol{w})),$$
$$\mu(\boldsymbol{w}) = \sum_{j=1}^{4} \mu_j(w_j) \quad \text{and} \quad \sigma(\boldsymbol{w}) = \prod_{j=1}^{4} \sigma_j(w_j),$$

where functions $\mu_j, \sigma_j, j = 1, \ldots, 4$, are specified below:

1. $\mu_1(w_1) = 1.5w_1, \sigma_1(w_1) = \left\{N(w_1, \mu = 0.2, \sigma^2 = 0.004) + N(w_1, \mu = 0.6, \sigma^2 = 0.1)\right\}/2,$
2. $\mu_2(w_2) = \left\{N(w_2, \mu = 0.2, \sigma^2 = 0.004) + N(w_2, \mu = 0.6, \sigma^2 = 0.1)\right\}/2,$
   $\sigma_2(w_2) = 0.6 + 0.5\sin(2\pi w_2),$
3. $\mu_3(w_3) = 1 + \sin(2\pi w_3), \sigma_3(w_3) = 1.1 - w_3,$
4. $\mu_4(w_4) = -w_4, \sigma_4(w_4) = 0.2 + 1.5w_4.$

To the generated dataset we fit a model with mean and variance functions modeled as

$$\mu(\boldsymbol{w}) = \beta_0 + \sum_{k=1}^{4}\sum_{j=1}^{16} \beta_{kj}\phi_{kj}(w_k) \quad \text{and} \quad \log\{\sigma^2(\boldsymbol{w})\} = \alpha_0 + \sum_{k=1}^{4}\sum_{j=1}^{16} \alpha_{kj}\phi_{kj}(w_k).$$

Fitting the above model to the simulated data is achieved by the following R code

```
> Model <- y ~ sm(w1, k = 15, bs = "rd") + sm(w2, k = 15, bs = "rd") +
+              sm(w3, k = 15, bs = "rd") + sm(w4, k = 15, bs = "rd") |
+              sm(w1, k = 15, bs = "rd") + sm(w2, k = 15, bs = "rd") +
+              sm(w3, k = 15, bs = "rd") + sm(w4, k = 15, bs = "rd")
```

```
> m3 <- mvrm(formula = Model, data = data, sweeps = 50000, burn = 25000,
+            thin = 5, seed = 1, StorageDir = DIR)
```

By default the function `sm` utilizes the radial basis functions, hence there is no need to specify `bs = "rd"`, as we did earlier, if radial basis functions are preferred over thin plate splines. Further, we have selected `k = 15` for all smooth functions. However, there is no restriction to the number of knots and certainly one can select a different number of knots for each smooth function.

As discussed previously, for each term that appears in the right-hand side of the mean and variance functions, the model incorporates indicator variables that specify which basis functions are to be included and which are to be excluded from the model. For the current example, the indicator variables are denoted by γ_{kj} and $\delta_{kj}, k = 1,2,3,4, j = 1,\ldots,16$. The prior probabilities that variables are included were specified in (8) and they are specific to each term, $\pi_{\mu_k} \sim \text{Beta}(c_{\mu_k}, d_{\mu_k})$, $\pi_{\sigma_k} \sim \text{Beta}(c_{\sigma_k}, d_{\sigma_k}), k = 1,2,3,4$. The default option `pi.muPrior = "Beta(1,1)"` specifies that $\pi_{\mu_k} \sim \text{Beta}(1,1), k = 1,2,3,4$. Further, by setting, for example, `pi.muPrior = "Beta(2,1)"` we specify that $\pi_{\mu_k} \sim \text{Beta}(2,1), k = 1,2,3,4$. To specify a different Beta prior for each of the four terms, `pi.muPrior` will have to be specified as a vector of length four, as an example, `pi.muPrior = c("Beta(1,1)","Beta(2,1)","Beta(3,1)","Beta(4,1)")`. Specification of the priors for π_{σ_k} is done in a similar way, via argument `pi.sigmaPrior`.

We conclude this section by presenting plots of the four terms in the mean and variance models. The plots are presented in Figure 5. We provide a few details on how the `plot` method works in the presence of multiple terms, and how the comparison between true and estimated effects is made. Starting with the mean function, to create the relevant plots, that appear on the left panels of Figure 5, the `plot` method considers only the part of the mean function $\mu(\boldsymbol{u})$ that is related to the chosen `term` while leaving all other terms out. For instance, in the code below we choose `term = "sm(u1)"` and hence we plot the posterior mean and a posterior credible interval for $\sum_{j=1}^{16} \beta_{1j}\phi_{1j}(u_1)$, where the intercept β_0 is left out by option `intercept = FALSE`. Further, comparison is made with a centered version of the true curve, represented by the dashed (red color) line and obtained by the first three lines of code below.

```
> x1 <- seq(0, 1, length.out = 30)
> y1 <- mu1(x1)
> y1 <- y1 - mean(y1)
> PlotOptions <- list(geom_line(aes_string(x = x1, y = y1),
+                     col = 2, alpha = 0.5, lty = 2))
> plot(x = m3, model = "mean", term = "sm(w1)", plotOptions = PlotOptions,
+      intercept = FALSE, centreEffects = FALSE, quantiles = c(0.005, 1 - 0.005))
```

The plots of the four standard deviation terms are shown in the right panels of Figure 5. Again, these are created by considering only the part of the model for $\sigma(\boldsymbol{u})$ that is related to the chosen `term`. For instance, below we choose `term = "sm(u1)"`. Hence, in this case the plot will present the posterior mean and a posterior credible interval for $\exp\{\sum_{j=1}^{16} \alpha_{1j}\phi_{1j}(u_1)/2\}$, where the intercept α_0 is left out by option `intercept = FALSE`. Option `centreEffects = TRUE` scales the posterior realizations of $\exp\{\sum_{j=1}^{16} \alpha_{1j}\phi_{1j}(u_1)/2\}$ before plotting them, where the scaling is done in such a way that the realized function has mean one over the range of the predictor. Further, the comparison is made with a scaled version of the true curve, where again the scaling is done to achieve mean one. This is shown below and it is in the spirit of Chan et al. (2006) who discuss the differences between the data generating mechanism and the fitted model.

```
> y1 <- stdev1(x1) / mean(stdev1(x1))
> PlotOptions <- list(geom_line(aes_string(x = x1, y = y1),
+                     col = 2, alpha = 0.5, lty = 2))
> plot(x = m3, model = "stdev", term = "sm(w1)", plotOptions = PlotOptions,
+      intercept = FALSE, centreEffects = TRUE, quantiles = c(0.025, 1 - 0.025))
```

Data analyses

In this section we present four empirical applications.

Wage and age

In the first empirical application, we analyse a dataset from Pagan and Ullah (1999) that is available in the R package **np** (Hayfield and Racine, 2008). The dataset consists of $n = 205$ observations on dependent variable `logwage`, the logarithm of the individual's wage, and covariate age, the individual's

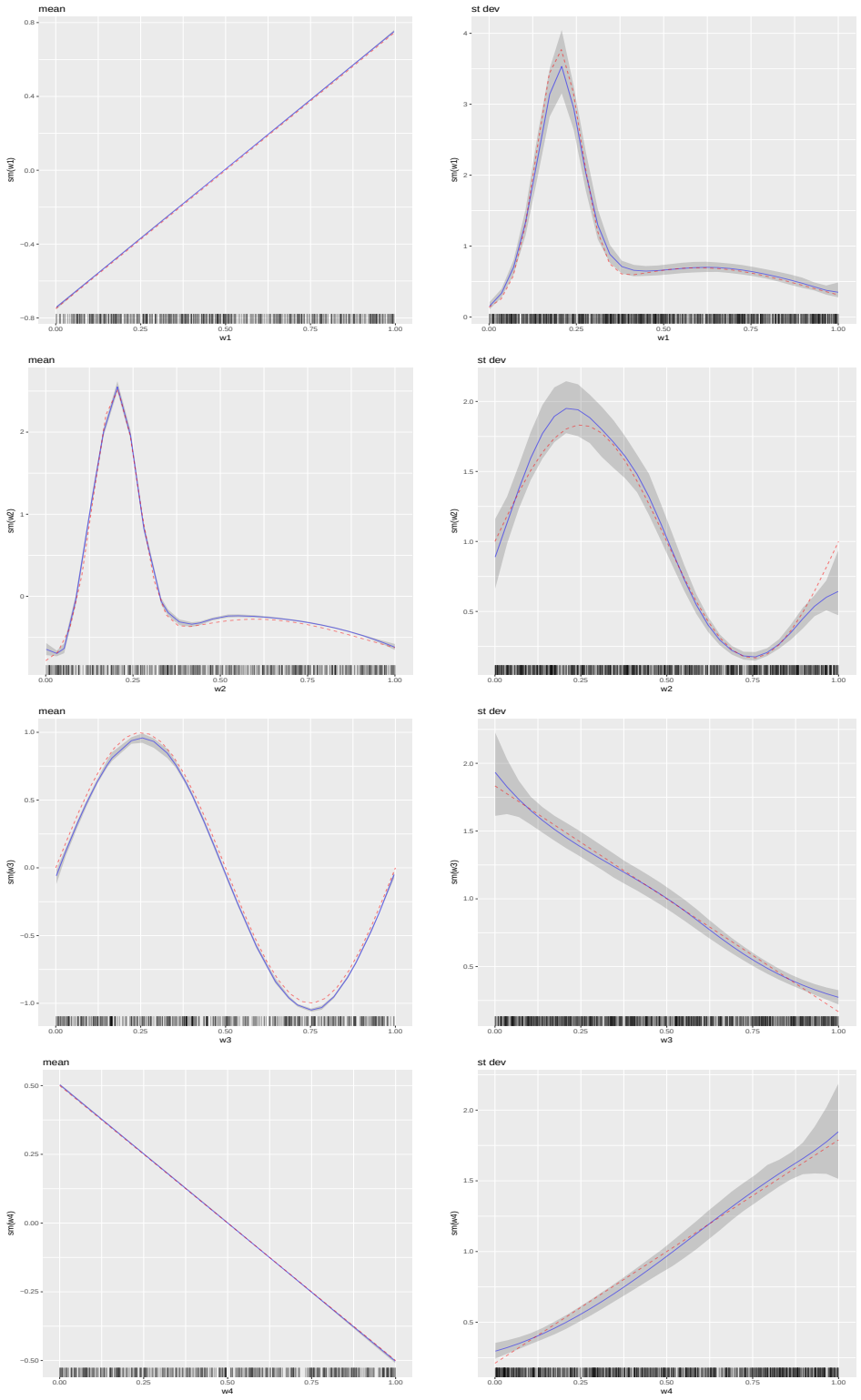
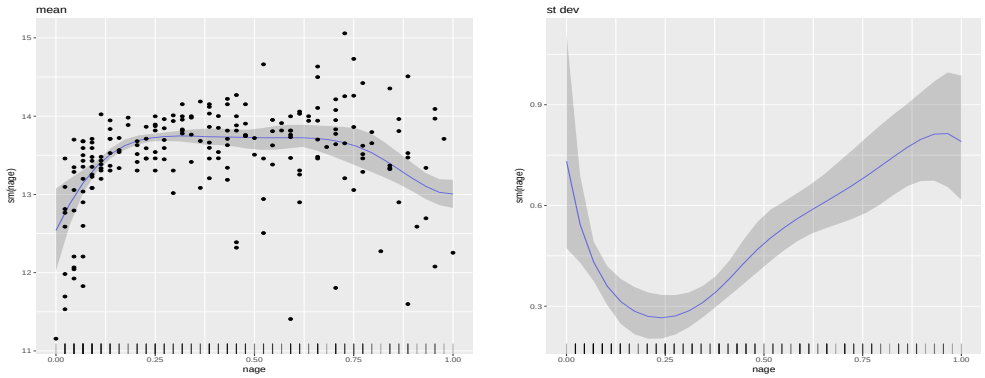


Figure 5: Multiple covariate simulation study results. The column on the left-hand side presents the true and estimated mean functions, along with 99% credible intervals. The column on the right-hand side presents the true and estimated standard deviation functions, along with 95% credible intervals. In all panels, the truth is represented by dashed (red color) lines, the estimated functions by solid (blue color) lines, and the credible intervals by gray color.



(a) (b)

Figure 6: Results from the data analysis on the relationship between age and the logarithm of wage. Panel (a) shows the posterior mean, an 80% credible interval of the mean function, and the observed data-points. Panel (b) shows the posterior mean and an 80% credible interval of the standard deviation function.

age. The dataset comes from the 1971 Census of Canada Public Use Sample Tapes and the sampling units it involves are males of common education. Hence, the investigation of the relationship between age and the logarithm of wage is carried out controlling for the two potentially important covariates education and gender.

We utilize the following R code to specify flexible models for the mean and variance functions, and to obtain 5,000 posterior samples, after a burn-in period of 25,000 samples and a thinning period of 5.

```
> data(cps71)
> DIR <- getwd()
> model <- logwage ~ sm(age, k = 30, bs = "rd") | sm(age, k = 30, bs = "rd")
> m4 <- mvrm(formula = model, data = cps71, sweeps = 50000,
+            burn = 25000, thin = 5, seed = 1, StorageDir = DIR)
```

After checking convergence, we use the following code to create the plots that appear in Figure 6.

```
> wagePlotOptions <- list(geom_point(data = cps71, aes(x = age, y = logwage)))
> plot(x = m4, model = "mean", term = "sm(age)", plotOptions = wagePlotOptions)
> plot(x = m4, model = "stdev", term = "sm(age)")
```

Figure 6 (a) shows the posterior mean and an 80% credible interval for the mean function and it suggests a quadratic relationship between age and `logwage`. Figure 6 (b) shows the posterior mean and an 80% credible interval for the standard deviation function. It suggest a complex relationship between age and the variability in `logwage`. The relationship suggested by Figure 6 (b) is also suggested by the spread of the data-points around the estimated mean in Figure 6 (a). At ages around 20 years the variability in `logwage` is high. It then reduces until about age 30, to start increasing again until about age 45. From age 45 to 60 it remains stable but high, while for ages above 60, Figure 6 (b) suggests further increase in the variability, but the wide credible interval suggests high uncertainty over this age range.

Wage and multiple covariates

In the second empirical application, we analyse a dataset from Wooldridge (2008) that is also available in the R package **np**. The response variable here is the logarithm of the individual's hourly wage (`lwage`) while the covariates include the years of education (`educ`), the years of experience (`exper`), the years with the current employer (`tenure`), the individual's gender (named as `female` within the dataset, with levels `Female` and `Male`), and marital status (named as `married` with levels `Married` and `Notmarried`). The dataset consists of $n = 526$ independent observations. We analyse the first three covariates as continuous and the last two as discrete.

As the variance function is modeled in terms of an exponential, see (1), to avoid potential numerical problems, we transform the three continuous variables to have range in the interval $[0, 1]$, using

```
> data(wage1)
> wage1$ntenure <- wage1$tenure / max(wage1$tenure)
```

```
> wage1$nexper <- wage1$exper / max(wage1$exper)
> wage1$neduc <- wage1$educ / max(wage1$educ)
```

We choose to fit the following mean and variance models to the data

$$\mu_i = \beta_0 + \beta_1\, \texttt{married}_i + f_1(\texttt{ntenure}_i) + f_2(\texttt{neduc}_i) + f_3(\texttt{nexper}_i, \texttt{female}_i),$$
$$\log(\sigma_i^2) = \alpha_0 + f_4(\texttt{nexper}_i).$$

We note that, as it turns out, an interaction between variables `nexper` and `female` is not necessary for the current data analysis. However, we choose to add this term in the mean model in order to illustrate how interaction terms can be specified. We illustrate further options below.

```
> knots1 <- seq(min(wage1$nexper), max(wage1$nexper), length.out = 30)
> knots2 <- c(0, 1)
> knotsD <- expand.grid(knots1, knots2)
> model <- lwage ~ fmarried + sm(ntenure) + sm(neduc, knots=data.frame(knots =
+ seq(min(wage1$neduc), max(wage1$neduc), length.out = 15))) +
+ sm(nexper, ffemale, knots = knotsD) | sm(nexper, knots=data.frame(knots =
+ seq(min(wage1$nexper), max(wage1$nexper), length.out=15)))
```

The first three lines of the R code above specify the matrix of (potential) knots to be used for representing $f_3(\texttt{nexper}, \texttt{female})$. Knots may be left unspecified, in which case the defaults in function `sm` will be used. Furthermore, in the specification of the mean model we use `sm(ntenure)`. By this, we chose to represent f_1 utilizing the default 10 knots and the radial basis functions. Further, the specification of f_2 in the mean model illustrates how users can specify their own knots for univariate functions. In the current example, we select 15 knots uniformly spread over the range of `neduc`. Fifteen knots are also used to represent f_4 within the variance model.

The following code is used to obtain samples from the posteriors of the model parameters.

```
> DIR <- getwd()
> m5 <- mvrm(formula = model, data = wage1, sweeps = 100000,
+            burn = 25000, thin = 5, seed = 1, StorageDir = DIR))
```

After summarizing results and checking convergence, we create plots of posterior means, along with 95% credible intervals, for functions $f_1, \ldots, f_4$. These are displayed in Figure 7. As it turns out, variable `married` does not have an effect on the mean of `lwage`. For this reason, we do not provide further results on the posterior of the coefficient of covariate `married`, β_1. However, in the code below we show how graphical summaries on β_1 can be obtained, if needed.

```
> PlotOptionsT <- list(geom_point(data = wage1, aes(x = ntenure, y = lwage)))
> plot(x = m5, model = "mean", term="sm(ntenure)", quantiles = c(0.025, 0.975),
+      plotOptions = PlotOptionsT)
> PlotOptionsEdu <- list(geom_point(data = wage1, aes(x = neduc, y = lwage)))
> plot(x = m5, model = "mean", term = "sm(neduc)", quantiles = c(0.025, 0.975),
+      plotOptions = PlotOptionsEdu)
> pchs <- as.numeric(wage1$female)
> pchs[pchs == 1] <- 17; pchs[pchs == 2] <- 19
> cols <- as.numeric(wage1$female)
> cols[cols == 2] <- 3; cols[cols == 1] <- 2
> PlotOptionsE <- list(geom_point(data = wage1, aes(x = nexper, y = lwage),
+                      col = cols, pch = pchs, group = wage1$female))
> plot(x = m5, model = "mean", term="sm(nexper,female)", quantiles = c(0.025, 0.975),
+      plotOptions = PlotOptionsE)
> plot(x = m5, model = "stdev", term = "sm(nexper)", quantiles = c(0.025, 0.975))
> PlotOptionsF <- list(geom_boxplot(fill = 2, color = 1))
> plot(x = m5, model = "mean", term = "married", quantiles = c(0.025, 0.975),
+      plotOptions = PlotOptionsF)
```

Figure 7, panels (a) and (b) show the posterior means and 95% credible intervals for $f_1(\texttt{ntenure})$ and $f_2(\texttt{neduc})$. It can be seen that expected wages increase with tenure and education, although there is high uncertainty over a large part of the range of both covariates. Panel (c) displays the posterior mean and a 95% credible interval for f_3. We can see that although the forms of the two functions are similar, (i.e. the interaction term is not needed), males have higher expected wages than females. Lastly, panel (d) displays posterior summaries of the standard deviation function, $\sigma_i = \sigma \exp(f_4/2)$. It can be seen that variability first increases and then decreases as experience increases.

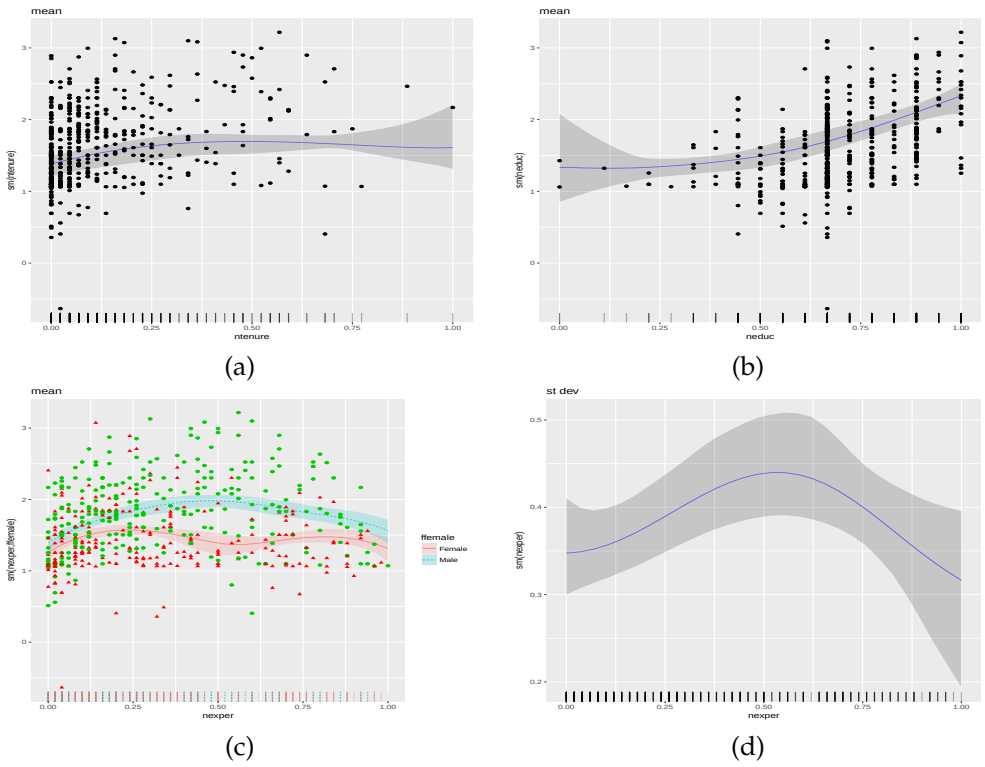


Figure 7: Results from the data analysis on the relationship between covariates gender, marital status, experience, education, and tenure, and response variable logarithm of hourly wage. Posterior means and 95% credible intervals for (a) $f_1(\texttt{ntenure})$, (b) $f_2(\texttt{neduc})$, (c) $f_3(\texttt{nexper}, \texttt{female})$, and (d) the standard deviation function $\sigma_i = \sigma \exp[f_4(\texttt{nexper})/2]$.

Lastly, we show how to obtain predictions and credible intervals for the levels `"Married"` and `"Notmarried"` of variable `fmaried` and the levels `"Female"` and `"Male"` of variable `fffemale`, with variables `ntenure`, `nedc`, and `nexper` fixed at their mid-range.

```
> p1 <- predict(m5, newdata = data.frame(fmarried = rep(c("Married", "Notmarried"), 2),
+       ntenure = rep(0.5, 4), neduc = rep(0.5, 4), nexper = rep(0.5, 4),
+       ffemale = rep(c("Female", "Male"), each = 2)), interval = "credible")

> p1
       fit      lwr      upr
1 1.321802 1.119508 1.506574
2 1.320400 1.119000 1.505272
3 1.913341 1.794035 2.036255
4 1.911939 1.791578 2.034832
```

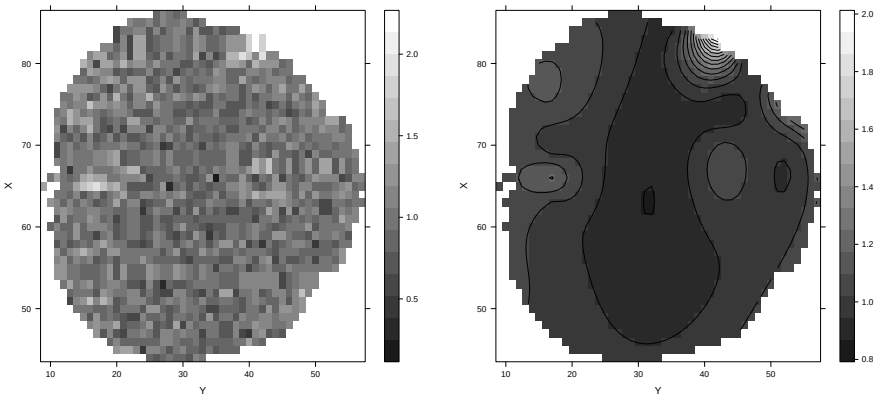
The predictions are suggestive of no "marriage" effect and of "gender" effect.

Brain activity

Here we analyse brain activity level data obtained by functional magnetic resonance imaging. The dataset is available in package **gamair** (Wood, 2006) and it was previously analysed by Landau et al. (2003). We are interested in three of the columns in the dataset. These are the response variable, `medFPQ`, which is calculated as the median over three measurements of "Fundamental Power Quotient" and the two covariates, `X` and `Y`, which show the location of each voxel.

The following R code loads the relevant data frame, removes two outliers and transforms the response variable, as was suggested by Wood (2006). In addition, it plots the brain activity data using function `levelplot` from the package **lattice** (Sarkar, 2008).

```
> data(brain)
> brain <- brain[brain$medFPQ > 5e-5, ]
> brain$medFPQ <- (brain$medFPQ) ^ 0.25
```



(a) (b)

Figure 8: Results from the brain activity level data analysis. Panel (a) shows the observed data and panel (b) the model-based smooth surface.

```
> levelplot(medFPQ ~ Y * X, data = brain, xlab = "Y", ylab = "X",
+           col.regions = gray(10 : 100 / 100))
```

The plot of the observed data is shown in Figure 8, panel (a). Its distinctive feature is the noise level, which makes it difficult to decipher any latent pattern. Hence, the goal of the current data analysis is to obtain a smooth surface of brain activity level from the noisy data. It was argued by Wood (2006) that for achieving this goal a spatial error term is not needed in the model. Thus, we analyse the brain activity level data using a model of the form

$$\text{medFPQ}_i \overset{\text{ind}}{\sim} N(\mu_i, \sigma^2), \text{ where } \mu_i = \beta_0 + \sum_{j_1=1}^{10} \sum_{j_2=1}^{10} \beta_{j_1,j_2} \phi_{1j_1,j_2}(X_i, Y_i), i = 1, \ldots, n,$$

where $n = 1565$ is the number of voxels.

The R code that fits the above model is

```
> Model <- medFPQ ~ sm(Y, X, k = 10, bs = "rd") | 1
> m6 <- mvrm(formula = Model, data = brain, sweeps = 50000, burn = 20000, thin = 2,
+            seed = 1, StorageDir = DIR)
```

From the fitted model we obtain a smooth brain activity level surface using function `predict`. The function estimates the average activity at each voxel of the brain. Further, we plot the estimated surface using function `levelplot`.

```
> p1 <- predict(m6)
> levelplot(p1[, 1] ~ Y * X, data = brain , xlab = "Y", ylab = "X",
+           col.regions = gray(10 : 100 / 100), contour = TRUE)
```

Results are shown in Figure 8, panel(b). The smooth surface makes it much easier to see and understand which parts of the brain have higher activity.

Cars

In the fourth and final application we use the function `mvrm` to identify the best subset of predictors in a regression setting. Usually stepwise model selection is performed, using functions `step` from base R and `stepAIC` from the **MASS** package. Here we show how `mvrm` can be used as an alternative to those two functions. The data frame that we apply `mvrm` on is `mtcars`, where the response variable is `mpg` and the explanatory variables that we consider are `disp`, `hp`, `wt`, and `qsec`. The code below loads the data frame, specifies the model and obtains samples from the posteriors of the model parameters.

```
> data(mtcars)
> Model <- mpg ~ disp + hp + wt + qsec | 1
> m7 <- mvrm(formula = Model, data = mtcars, sweeps = 50000, burn = 25000, thin = 2,
+            seed = 1, StorageDir = DIR)
```

The following is an excerpt of the output that the `summary` method produces showing the three models with the highest posterior probability.

```
> summary(m7, nModels = 3)

Joint mean/variance model posterior probabilities:
mean.disp mean.hp mean.wt mean.qsec freq  prob cumulative
1       0       1       1         0 1085 43.40      43.40
2       0       0       1         1 1040 41.60      85.00
3       0       0       1         0  128  5.12      90.12
Displaying 3 models of the 11 visited
3 models account for 90.12% of the posterior mass
```

The model with the highest posterior probability (43.4%) is the one that includes explanatory variables `hp` and `wt`. The model that includes `wt` and `qsec` has almost equal posterior probability, 41.6%. These two models account for 85% of the posterior mass. The third most promising model is the one that includes only `wt` as predictor, but its posterior probability is much lower, 5.12%.

Appendix: MCMC algorithm

In this section we present the technical details of how the MCMC algorithm is designed for the case where there is a single covariate in the mean and variance models. We first note that to improve mixing of the sampler, we integrate out vector $\boldsymbol{\beta}$ from the likelihood of $\boldsymbol{y}$, as was done by Chan et al. (2006):

$$f(\boldsymbol{y}|\boldsymbol{\alpha}, c_\beta, \boldsymbol{\gamma}, \boldsymbol{\delta}, \sigma^2) \propto |\sigma^2 D^2(\boldsymbol{\alpha}_\delta)|^{-\frac{1}{2}} (c_\beta + 1)^{-\frac{N(\gamma)+1}{2}} \exp(-S/2\sigma^2), \tag{10}$$

where, with $\tilde{\boldsymbol{y}} = \boldsymbol{D}^{-1}(\boldsymbol{\alpha}_\delta)\boldsymbol{y}$, we have $S = S(\boldsymbol{y}, \boldsymbol{\alpha}, c_\beta, \boldsymbol{\gamma}, \boldsymbol{\delta}) = \tilde{\boldsymbol{y}}^\top \tilde{\boldsymbol{y}} - \frac{c_\beta}{1+c_\beta} \tilde{\boldsymbol{y}}^\top \tilde{\boldsymbol{X}}_\gamma (\tilde{\boldsymbol{X}}_\gamma^\top \tilde{\boldsymbol{X}}_\gamma)^{-1} \tilde{\boldsymbol{X}}_\gamma^\top \tilde{\boldsymbol{y}}$.

The six steps of the MCMC sampler are as follows

1. Similar to Chan et al. (2006), the elements of $\boldsymbol{\gamma}$ are updated in blocks of randomly chosen elements. The block size is chosen based on probabilities that can be supplied by the user or be left at their default values. Let $\boldsymbol{\gamma}_B$ be a block of random size of randomly chosen elements from $\boldsymbol{\gamma}$. The proposed value for $\boldsymbol{\gamma}_B$ is obtained from its prior with the remaining elements of $\boldsymbol{\gamma}$, denoted by $\boldsymbol{\gamma}_{B^C}$, kept at their current value. The proposal pmf is obtained from the Bernoulli prior with π_μ integrated out

$$p(\boldsymbol{\gamma}_B|\boldsymbol{\gamma}_{B^C}) = \frac{p(\boldsymbol{\gamma})}{p(\boldsymbol{\gamma}_{B^C})} = \frac{\text{Beta}(c_\mu + N(\boldsymbol{\gamma}), d_\mu + q_1 - N(\boldsymbol{\gamma}))}{\text{Beta}(c_\mu + N(\boldsymbol{\gamma}_{B^C}), d_\mu + q_1 - L(\boldsymbol{\gamma}_B) - N(\boldsymbol{\gamma}_{B^C}))},$$

   where $L(\boldsymbol{\gamma}_B)$ denotes the length of $\boldsymbol{\gamma}_B$ (i.e., the size of the block). For this proposal pmf, the acceptance probability of the Metropolis-Hastings move reduces to the ratio of the likelihoods in (10)

$$\min\left\{1, \frac{(c_\beta+1)^{-\frac{N(\boldsymbol{\gamma}^P)+1}{2}} \exp(-S^P/2\sigma^2)}{(c_\beta+1)^{-\frac{N(\boldsymbol{\gamma}^C)+1}{2}} \exp(-S^C/2\sigma^2)}\right\},$$

   where superscripts P and C denote proposed and currents values respectively.

2. Vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ are updated simultaneously. Similarly to the updating of $\boldsymbol{\gamma}$, the elements of $\boldsymbol{\delta}$ are updated in random order in blocks of random size. Let $\boldsymbol{\delta}_B$ denote a block. Blocks $\boldsymbol{\delta}_B$ and the whole vector $\boldsymbol{\alpha}$ are generated simultaneously. As was mentioned by Chan et al. (2006), generating the whole vector $\boldsymbol{\alpha}$, instead of subvector $\boldsymbol{\alpha}_B$, is necessary in order to make the proposed value of $\boldsymbol{\alpha}$ consistent with the proposed value of $\boldsymbol{\delta}$.

   Generating the proposed value for $\boldsymbol{\delta}_B$ is done in a similar way as was done for $\boldsymbol{\gamma}_B$ in the previous step. Let $\boldsymbol{\delta}^P$ denote the proposed value of $\boldsymbol{\delta}$. Next, we describe how the proposed vale for $\boldsymbol{\alpha}_{\delta^P}$ is obtained. The development is in the spirit of Chan et al. (2006) who built on the work of Gamerman (1997).

   Let $\hat{\boldsymbol{\beta}}_\gamma^C = \{c_\beta/(1+c_\beta)\}(\tilde{\boldsymbol{X}}_\gamma^\top \tilde{\boldsymbol{X}}_\gamma)^{-1} \tilde{\boldsymbol{X}}_\gamma^\top \tilde{\boldsymbol{y}}$ denote the current value of the posterior mean of $\boldsymbol{\beta}_\gamma$. Define the current squared residuals

$$e_i^C = (y_i - (\boldsymbol{x}_{i\gamma}^*)^\top \hat{\boldsymbol{\beta}}_\gamma^C)^2,$$

   $i = 1, \ldots, n$. These will have an approximate $\sigma_i^2 \chi_1^2$ distribution, where $\sigma_i^2 = \sigma^2 \exp(\boldsymbol{z}_i^\top \boldsymbol{\alpha})$. The

latter defines a Gamma generalized linear model (GLM) for the squared residuals with mean $E(\sigma_i^2\chi_1^2) = \sigma_i^2 = \sigma^2\exp(\boldsymbol{z}_i^\top\boldsymbol{\alpha})$, which, utilizing a log-link, can be thought of as Gamma GLM with an offset term: $\log(\sigma_i^2) = \log(\sigma^2) + \boldsymbol{z}_i^\top\boldsymbol{\alpha}$. Given the proposed value of $\boldsymbol{\delta}$, denoted by $\boldsymbol{\delta}^P$, the proposal density for $\boldsymbol{\alpha}_{\delta^P}^P$ is derived utilizing the one step iteratively re-weighted least squares algorithm. This proceeds as follows. First define the transformed observations

$$d_i^C(\boldsymbol{\alpha}^C) = \log(\sigma^2) + \boldsymbol{z}_i^\top\boldsymbol{\alpha}^C + \frac{e_i^C - (\sigma_i^2)^C}{(\sigma_i^2)^C},$$

where superscript C denotes current values. Further, let $\boldsymbol{d}^C$ denote the vector of d_i^C.
Next we define

$$\boldsymbol{\Delta}(\boldsymbol{\delta}^P) = (c_\alpha^{-1}\boldsymbol{I} + \boldsymbol{Z}_{\delta^P}^\top\boldsymbol{Z}_{\delta^P})^{-1} \text{ and} \hat{\boldsymbol{\alpha}}(\boldsymbol{\delta}^P, \boldsymbol{\alpha}^C) = \boldsymbol{\Delta}_{\delta^P}\boldsymbol{Z}_{\delta^P}^\top\boldsymbol{d}^C,$$

where $\boldsymbol{Z}$ is the design matrix. The proposed value $\boldsymbol{\alpha}_{\delta^P}^P$ is obtained from a multivariate normal distribution with mean$\hat{\boldsymbol{\alpha}}(\boldsymbol{\delta}^P, \boldsymbol{\alpha}^C)$ and covariance $h\boldsymbol{\Delta}(\boldsymbol{\delta}^P)$, denoted as $N(\boldsymbol{\alpha}_{\delta^P}^P; \hat{\boldsymbol{\alpha}}(\boldsymbol{\delta}^P, \boldsymbol{\alpha}^C), h\boldsymbol{\Delta}(\boldsymbol{\delta}^P))$, where h is a free parameter that we introduce and select its value adaptively (Roberts and Rosenthal, 2009) in order to achieve an acceptance probability of $20\% - 25\%$ (Roberts and Rosenthal, 2001).
Let $N(\boldsymbol{\alpha}_{\delta^C}^C; \hat{\boldsymbol{\alpha}}(\boldsymbol{\delta}^C, \boldsymbol{\alpha}^P), h\boldsymbol{\Delta}(\boldsymbol{\delta}^C))$ denote the proposal density for taking a step in the reverse direction, from model $\boldsymbol{\delta}^P$ to $\boldsymbol{\delta}^C$. Then the acceptance probability of the pair $(\boldsymbol{\delta}^P, \boldsymbol{\alpha}_{\delta^P}^P)$ is the minimum between 1 and

$$\frac{|D^2(\boldsymbol{\alpha}_{\delta^P}^P)|^{-\frac{1}{2}}\exp(-S^P/2\sigma^2)}{|D^2(\boldsymbol{\alpha}_{\delta^C}^C)|^{-\frac{1}{2}}\exp(-S^C/2\sigma^2)}\frac{(2\pi c_\alpha)^{-\frac{N(\delta^P)}{2}}\exp(-\frac{1}{2c_\alpha}(\boldsymbol{\alpha}_{\delta^P}^P)^\top\boldsymbol{\alpha}_{\delta^P}^P)}{(2\pi c_\alpha)^{-\frac{N(\delta^C)}{2}}\exp(-\frac{1}{2c_\alpha}(\boldsymbol{\alpha}_{\delta^C}^C)^\top\boldsymbol{\alpha}_{\delta^C}^C)}\frac{N(\boldsymbol{\alpha}_{\delta^C}^C; \hat{\boldsymbol{\alpha}}_{\delta^C}, h\boldsymbol{\Delta}_{\delta^C})}{N(\boldsymbol{\alpha}_{\delta^P}^P; \hat{\boldsymbol{\alpha}}_{\delta^P}, h\boldsymbol{\Delta}_{\delta^P})}.$$

We note that the determinants that appear in the above ratio are equal to one when utilizing centred explanatory variables in the variance model and hence can be left out of the calculation of the acceptance probability.

3. We update σ^2 utilizing the marginal (10) and the two priors in (7). The full conditional corresponding to the IG(a_σ, b_σ) prior is

$$f(\sigma^2|\dots) \propto (\sigma^2)^{-\frac{n}{2}-a_\sigma-1}\exp\{-(S/2+b_\sigma)/\sigma^2\},$$

which is recognized as another inverse gamma IG$(n/2+a_\sigma, S/2+b_\sigma)$ distribution.
The full conditional corresponding to the normal prior $|\sigma| \sim N(0, \phi_\sigma^2)$ is

$$f(\sigma^2|\dots) \propto (\sigma^2)^{-\frac{n}{2}}\exp(-S/2\sigma^2)\exp(-\sigma^2/2\phi_\sigma^2).$$

Proposed values are obtained from $\sigma_p^2 \sim N(\sigma_c^2, f^2)$ where σ_c^2 denotes the current value. Proposed values are accepted with probability $f(\sigma_p^2|\dots)/f(\sigma_c^2|\dots)$. We treat f^2 as a tuning parameter and we select its value adaptively (Roberts and Rosenthal, 2009) in order to achieve an acceptance probability of $20\% - 25\%$ (Roberts and Rosenthal, 2001).

4. Parameter c_β is updated from the marginal (10) and the IG(a_β, b_β) prior

$$f(c_\beta|\dots) \propto (c_\beta+1)^{-\frac{N(\gamma)+1}{2}}\exp(-S/2\sigma^2)(c_\beta)^{-a_\beta-1}\exp(-b_\beta/c_\beta).$$

To sample from the above, we utilize a normal approximation to it. Let $\ell(c_\beta) = \log\{f(c_\beta|\dots)\}$. We utilize a normal proposal density $N(\hat{c}_\beta, -g^2/\ell''(\hat{c}_\beta))$, where $\hat{c}_\beta$ is the mode of $\ell(c_\beta)$, found using a Newton-Raphson algorithm, $\ell''(\hat{c}_\beta)$ is the second derivative of $\ell(c_\beta)$ evaluated at the mode, and g^2 is a tuning variance parameter the value of which is chosen adaptively (Roberts and Rosenthal, 2009). At iteration $u+1$ the acceptance probability is the minimum between one and

$$\frac{f(c_\beta^{(u+1)}|\dots)}{f(c_\beta^{(u)}|\dots)}\frac{N(c_\beta^{(u)}; \hat{c}_\beta, -g^2/\ell''(\hat{c}_\beta))}{N(c_\beta^{(u+1)}; \hat{c}_\beta, -g^2/\ell''(\hat{c}_\beta))}.$$

5. Parameter c_α is updated from the inverse Gamma density IG$(a_\alpha + N(\delta)/2, b_\alpha + \boldsymbol{\alpha}_\delta^\top\boldsymbol{\alpha}_\delta/2)$.
6. The sampler utilizes the marginal in (10) to improve mixing. However, if samples are required from the posterior of $\boldsymbol{\beta}$, they can be generated from

$$\boldsymbol{\beta}_\gamma|\dots \sim N(\frac{c_\beta}{1+c_\beta}(\tilde{\boldsymbol{X}}_\gamma^\top\tilde{\boldsymbol{X}}_\gamma)^{-1}\tilde{\boldsymbol{X}}_\gamma^\top\tilde{\boldsymbol{y}}, \frac{\sigma^2 c_\beta}{1+c_\beta}(\tilde{\boldsymbol{X}}_\gamma^\top\tilde{\boldsymbol{X}}_\gamma)^{-1}),$$

where β_γ is the non-zero part of β.

Summary

We have presented a tutorial on several functions from the R package **BNSP**. These functions are used for specifying, fitting and summarizing results from regression models with Gaussian errors and with mean and variance functions that can be modeled nonparametrically. Function `sm` is utilized to specify smooth terms in the mean and variance functions of the model. Function `mvrm` calls an MCMC algorithm that obtains samples from the posteriors of the model parameters. Samples are converted into an object of class `"mcmc"` by the function `mvrm2mcmc` which facilitates the use of multiple functions from package **coda**. Functions `print.mvrm` and `summary.mvrm` provide summaries of fitted `"mvrm"` objects. Further, function `plot.mvrm` provides graphical summaries of parametric and nonparametric terms that enter the mean or variance function. Lastly, function `predict.mvrm` provides predictions for a future response or a mean response along with the corresponding prediction/credible intervals.

Bibliography

P.-C. Bürkner. **brms**: *Bayesian Regression Models Using 'Stan'*, 2018. URL https://CRAN.R-project.org/package=brms. R package version 2.3.1. [p526]

D. Chan, R. Kohn, D. Nott, and C. Kirby. Locally adaptive semiparametric estimation of the mean and variance functions in regression models. *Journal of Computational and Graphical Statistics*, 15(4): 915–936, 2006. ISSN 10618600. URL https://doi.org/10.1198/106186006X157441. [p526, 527, 528, 530, 538, 544]

D. Gamerman. Sampling from the posterior distribution in generalized linear mixed models. *Statistics and Computing*, 7(1):57–68, 1997. ISSN 1573-1375. URL https://doi.org/10.1023/A:1018509429360. [p544]

E. I. George and R. E. McCulloch. Variable selection via Gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993. URL https://doi.org/10.1080/01621459.1993.10476353. [p526]

E. I. George and R. E. McCulloch. Approaches for Bayesian variable selection. *Statistica Sinica*, 7(2): 339–373, 1997. [p526]

T. Hayfield and J. S. Racine. Nonparametric econometrics: The np package. *Journal of Statistical Software*, 27(5), 2008. URL http://dx.doi.org/10.18637/jss.v027.i05. [p538]

B. Hofner, A. Mayr, N. Fenske, and M. Schmid. *gamboostLSS: Boosting Methods for GAMLSS Models*, 2018. URL https://CRAN.R-project.org/package=gamboostLSS. R package version 2.0-1. [p526]

S. Landau, I. C. Ellison-Wright, and E. T. Bullmore. Tests for a difference in timing of physiological response between two brain regions measured by using functional magnetic resonance imaging. *Journal of the Royal Statistical Society C*, 53(1):63–82, 2003. URL https://doi.org/10.1111/j.0035-9254.2003.04844.x. [p542]

B. W. Lewis. *Threejs: Interactive 3D Scatter Plots, Networks and Globes*, 2016. URL https://CRAN.R-project.org/package=threejs. R package version 0.2.2. [p527]

F. Liang, R. Paulo, G. Molina, M. A. Clyde, and J. O. Berger. Mixtures of g Priors for Bayesian variable selection. *Journal of the American Statistical Association*, 103(481):410–423, 2008. ISSN 0162-1459, 1537-274X. URL https://doi.org/10.1198/016214507000001337. [p531]

A. Mayr, N. Fenske, B. Hofner, T. Kneib, and M. Schmid. Generalized additive models for location, scale and shape for high dimensional data—a flexible approach based on boosting. *Journal of the Royal Statistical Society C*, 61(3):403–427, 2012. URL https://doi.org/10.1111/j.1467-9876.2011.01033.x. [p527, 535]

P. Müller and R. Mitra. Bayesian nonparametric inference — why and how. *Bayesian Analysis*, 8(2): 269–302, 2013. URL https://doi.org/10.1214/13-BA811. [p526]

R. B. O'Hara and M. J. Sillanpää. A review of Bayesian variable selection methods: What, how and which. *Bayesian Analysis*, 4(1):85–117, 2009. URL https://doi.org/10.1214/09-BA403. [p526]

A. Pagan and A. Ullah. *Nonparametric Econometrics*. Cambridge University Press, 1999. URL https://doi.org/10.1017/CBO9780511612503. [p538]

G. Papageorgiou. *BNSP: Bayesian Non- And Semi-Parametric Model Fitting*, 2018. URL https://CRAN.R-project.org/package=BNSP. R package version 2.0.7. [p526]

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence Diagnosis and Output Analysis for MCMC. *R News*, 6(1):7–11, 2006. URL http://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf. [p527]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016. URL https://www.R-project.org/. ISBN 3-900051-07-0. [p526]

R. A. Rigby and D. M. Stasinopoulos. Generalized additive models for location, scale and shape, (with discussion). *Journal of the Royal Statistical Society C*, 54(3):507–554, 2005. URL https://doi.org/10.1111/j.1467-9876.2005.00510.x. [p526]

G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, 2001. URL https://doi.org/10.1214/ss/1015346320. [p545]

G. O. Roberts and J. S. Rosenthal. Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009. ISSN 1061-8600, 1537-2715. URL https://doi.org/10.1198/jcgs.2009.06134. [p527, 545]

D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer-Verlag, New York, 2008. URL http://lmdvr.r-forge.r-project.org. ISBN 978-0-387-75968-5. [p542]

F. Scheipl. spikeSlabGAM: Bayesian variable selection, model choice and regularization for generalized additive mixed models in R. *Journal of Statistical Software*, 43(14):1–24, 2011. URL http://dx.doi.org/10.18637/jss.v043.i14. [p526]

F. Scheipl. *spikeSlabGAM: Bayesian Variable Selection and Model Choice for Generalized Additive Mixed Models*, 2016. URL https://CRAN.R-project.org/package=spikeSlabGAM. R package version 1.1-11. [p526]

K. Soetaert. *plot3D: Plotting Multi-Dimensional Data*, 2016. URL https://CRAN.R-project.org/package=plot3D. R package version 1.1. [p527]

D. M. Stasinopoulos and R. A. Rigby. Generalized additive models for location scale and shape (GAMLSS) in R. *Journal of Statistical Software*, 23(7):1–46, 2007. URL http://dx.doi.org/10.18637/jss.v023.i07. [p526]

N. Umlauf, N. Klein, A. Zeileis, and M. Koehler. *bamlss: Bayesian Additive Models for Location Scale and Shape (and Beyond)*, 2017. URL https://CRAN.R-project.org/package=bamlss. R package version 0.1-2. [p526]

N. Umlauf, N. Klein, and A. Zeileis. BAMLSS: Bayesian additive models for location, scale and shape (and beyond). *Journal of Computational and Graphical Statistics*, 2018. URL http://dx.doi.org/10.1080/10618600.2017.1407325. [p526]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p527]

S. Wood. *Mgcv: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation*, 2018. URL https://CRAN.R-project.org/package=mgcv. R package version 1.8-23. [p527]

S. N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, Boca Raton, Florida, 2006. ISBN 1-58488-474-6. [p542, 543]

J. Wooldridge. *Introductory Econometrics: A Modern Approach*. South Western College, 2008. [p540]

A. Zeileis and Y. Croissant. Extended model formulas in R: Multiple parts and multiple responses. *Journal of Statistical Software*, 34(1):1–13, 2010. URL http://dx.doi.org/10.18637/jss.v034.i01. [p531]

A. Zeileis, K. Hornik, and P. Murrell. Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, 53(9):3259–3270, 2009. URL https://doi.org/10.1016/j.csda.2008.11.033. [p536]

A. Zellner. On assessing prior distributions and Bayesian regression analysis with g-prior distributions. In P. Goel and Zellner, editors, *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno De Finetti*, pages 233–243. Elsevier Science Publishers, 1986. [p528]

Georgios Papageorgiou
Department of Economics, Mathematics and Statistics
Birkbeck, University of London
g.papageorgiou@bbk.ac.uk