Journal of Statistical Software
MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

Semiparametric Regression Analysis via Infer.NET

Jan Luts
TheSearchParty.com Pty Ltd.

Shen S.J. Wang
University of Queensland

John T. Ormerod
University of Sydney

Matt P. Wand
University of Technology, Sydney

Abstract

We provide several examples of Bayesian semiparametric regression analysis via the Infer.NET package for approximate deterministic inference in graphical models. The examples are chosen to encompass a wide range of semiparametric regression situations. Infer.NET is shown to produce accurate inference in comparison with Markov chain Monte Carlo via the BUGS package, but is considerably faster. Potentially, this contribution represents the start of a new era for semiparametric regression, where large and complex analyses are performed via fast graphical models methodology and software, mainly being developed within Machine Learning.

Keywords: additive mixed models, expectation propagation, generalized additive models, measurement error models, mean field variational Bayes, missing data models, penalized splines, variational message passing .

Date of this draft: 17 JAN 2014

1. Introduction

Infer.NET (Minka *et al.* 2013) is a relatively new software package for performing approximate inference in large graphical models, using fast deterministic algorithms such as expectation propagation (Minka 2001) and variational message passing (Winn and Bishop 2005). We demonstrate its application to Bayesian semiparametric regression. A variety of situations are covered: non-Gaussian response, longitudinal data, bivariate functional effects, robustness, sparse signal penalties, missingness and measurement error.

The Infer.NET project is still in its early years and, and at the time of this writing, has not progressed beyond beta releases. We anticipate continual updating and enhancement for many years to come. In this article we are necessarily restricted to the capabilities of Infer.NET at the time of preparation. All of our examples use Infer.NET 2.5, Beta 2, which was released in April 2013.

The graphical models viewpoint of semiparametric regression (Wand 2009), and other advanced statistical techniques, has the attraction that various embellishments of the standard models can be accommodated by enlarging the underlying directed acyclic graph. For example, nonparametric regression with a missing predictor data model involves the addition of nodes and edges to the graph, corresponding to the missing data mechanism. Figure 4 of Faes *et al.* (2011) provides some specific illustrations. Marley and Wand (2010) exploited the graphical models viewpoint of semiparametric regression to handle a wide range of nonstandard models via the Markov chain Monte Carlo inference engine implemented by BUGS (Lunn *et al.* 2000). However, many of their examples take between several minutes and hours to run. The inference engines provided by Infer.NET allow much faster fitting for many common semiparametric regression models. On the other hand, BUGS is the more versatile package and not all models that are treated in Marley and Wand (2010) are supported by Infer.NET.

Semiparametric regression, summarized by Ruppert *et al.* (2003, 2009), is a large branch of Statistics that includes nonparametric regression, generalized additive models, generalized additive mixed models, curve-by-factor interaction models, wavelet regression and geoadditive models. Parametric models such as generalized linear mixed models are special cases of semiparametric regression. Antecedent research for special cases such as nonparametric regression was conducted in the second half of the 20th Century at a time when data sets were smaller, computing power was lower and the Internet was either non-existent or in its infancy. Now, as we approach the mid-2010s, semiparametric regression is continually being challenged by the size, complexity and, in some applications, arrival speed of data-sets requiring analysis. Implementation and computational speed are major limiting factors. This contribution represents the potential for a new era for semiparametric regression – tapping into 21st Century Machine Learning research on fast approximate inference on large graphical models (e.g. Minka 2001; Winn and Bishop 2005; Minka 2005; Minka and Winn 2008; Knowles and Minka 2011) and ensuing software development.

Section 2 lays down the definitions and notation needed to describe the models given in later sections. Each of Sections 3–10 illustrates a different type of semiparametric regression analysis via Infer.NET. All code is available as web-supplement to this articles. For most of the examples, we also compare the Infer.NET results with those produced by BUGS. Section 11 compares BUGS with Infer.NET in terms of versatility and computational speed. A summary of our findings on semiparametric analysis via Infer.NET is given in Section 12. An appendix gives a detailed description of using Infer.NET to fit the simple semiparametric regression model of Section 3.

2. Preparatory infrastructure

The semiparametric regression examples rely on mathematical infrastructure and notation, which we describe in this section.

2.1. Distributional notation

The density function of a random vector $\boldsymbol{x}$ is denoted by $p(\boldsymbol{x})$. The notation for a set of independent random variables y_i $(1 \leq i \leq n)$ with distribution D_i is $y_i \overset{\text{ind.}}{\sim} D_i$. Table 1 lists the distributions used in the examples and the parametrization of their density functions.

distribution	density function in x	abbreviation
Normal	$(2\pi\sigma^2)^{-1/2}\exp\{-(x-\mu)^2/(2\sigma^2)\}$; $\sigma > 0$	$N(\mu, \sigma^2)$
Laplace	$(2\sigma)^{-1}\exp(-\lvert x-\mu\rvert/\sigma)$; $\sigma > 0$	$\text{Laplace}(\mu, \sigma^2)$
t	$\dfrac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu\sigma^2}\Gamma(\nu/2)\{1+\frac{(x-\mu)^2}{\nu\sigma^2}\}^{\frac{\nu+1}{2}}}$; $\sigma, \nu > 0$	$t(\mu, \sigma^2, \nu)$
Gamma	$\dfrac{B^A\, x^{A-1} e^{-B\,x}}{\Gamma(A)}$; $x > 0$; $A, B > 0$	$\text{Gamma}(A, B)$
Half-Cauchy	$\dfrac{2\sigma}{\pi(x^2+\sigma^2)}$; $x > 0$; $\sigma > 0$	$\text{Half-Cauchy}(\sigma)$

Table 1: Distributions used in the examples. The density function argument x and parameters range over $\mathbb{R}$ unless otherwise specified.

2.2. Standardization and default priors

In real data examples, the measurements are often recorded on several different scales. Therefore, all continuous variables are standardized prior to fitting analysis using Infer.NET or Markov Chain Monte Carlo (MCMC) in BUGS. This transformation makes the analyses scale-invariant and can also lead to better behavior of the MCMC.

Since we do not have prior knowledge about the model parameters in each of the examples, non-informative priors are used. The prior distributions for a fixed effects parameter vector $\boldsymbol{\beta}$ and a standard deviation parameter σ are

$$\boldsymbol{\beta} \sim N(\mathbf{0}, \tau_\beta^{-1}\boldsymbol{I}) \quad \text{and} \quad \sigma \sim \text{Half-Cauchy}(A)$$

with default hyperparameters

$$\tau_\beta = 10^{-10} \quad \text{and} \quad A = 10^5.$$

This is consistent with the recommendations given in Gelman (2006) for achieving non-informativeness for variance parameters. Infer.NET and BUGS do not offer direct specification of Half-Cauchy distributions and therefore we use the result:

$$\begin{aligned} &\text{if} \quad x|\, a \sim \text{Gamma}(\tfrac{1}{2}, a) \quad \text{and} \quad a \sim \text{Gamma}(\tfrac{1}{2}, 1/A^2) \\ &\text{then} \quad x^{-1/2} \sim \text{Half-Cauchy}(A). \end{aligned} \tag{1}$$

This result allows for the imposition of a Half-Cauchy prior using only Gamma distribution specifications. Both Infer.NET and BUGS support the Gamma distribution.

2.3. Variational message passing and expectation propagation

Infer.NET has two inference engines, *variational message passing* (Winn and Bishop 2005) and *expectation propagation* (Minka 2001), for performing fast deterministic approximate inference in graphical models. Succinct summaries of variational message passing and expectation propagation are provided in Appendices A and B of Minka and Winn (2008).

Generally speaking, variational message passing is more amenable to semiparametric regression than expectation propagation. It is a special case of *mean field variational Bayes* (e.g. Wainwright and Jordan 2008). The essential idea of mean field variational Bayes is to approximate joint posterior density functions such as $p(\theta_1, \theta_2, \theta_3|\boldsymbol{D})$, where $\boldsymbol{D}$ denotes the observed data, by product density forms such as

$$q_{\theta_1}(\theta_1)\,q_{\theta_2}(\theta_2)\,q_{\theta_3}(\theta_3), \quad q_{\theta_1,\theta_3}(\theta_1,\theta_3)\,q_{\theta_2}(\theta_2) \quad \text{or} \quad q_{\theta_1}(\theta_1)\,q_{\theta_2,\theta_3}(\theta_2,\theta_3). \tag{2}$$

The choice of the product density form is usually made by trading off tractability against minimal imposition of product structure. Once this is choice is made, the optimal q-density functions are chosen to minimize the Kullback-Liebler divergence from the exact joint posterior density function. For example, if the second product form in (2) is chosen then the optimal density functions $q^*_{\theta_1,\theta_3}(\theta_1,\theta_3)$ and $q^*_{\theta_2}(\theta_2)$ are those that minimize

$$\int\!\!\int\!\!\int q_{\theta_1,\theta_3}(\theta_1,\theta_3)\,q_{\theta_2}(\theta_2)\log\left\{\frac{p(\theta_1,\theta_2,\theta_3|\,\boldsymbol{D})}{q_{\theta_1,\theta_3}(\theta_1,\theta_3)\,q_{\theta_2}(\theta_2)}\right\}\,d\theta_1\,d\theta_2\,d\theta_3, \tag{3}$$

where the integrals range over the parameter spaces of θ_1, θ_2 and θ_3. This minimization problem gives rise to an iterative scheme which, typically, has closed form updates and good convergence properties. A by-product of the iterations is a lower-bound approximation to the marginal likelihood $p(\boldsymbol{D})$, which we denote by $\underline{p}(\boldsymbol{D})$.

Further details on, and several examples of, mean field variational Bayes are provided by Section 2.2 of Ormerod and Wand (2010). Each of these examples can also be expressed, equivalently, in terms of variational message passing.

In typical semiparametric regression models the subscripting on the q-density functions is quite cumbersome. Hence, it is suppressed for the remainder of the article. For example, $q(\theta_1,\theta_3)$ is taken to mean $q_{\theta_1,\theta_3}(\theta_1,\theta_3)$.

Expectation propagation is also based on product density restrictions such as (2), but differs in its method of obtaining the optimal density functions. It works with a different version of the Kullback-Leibler divergence than that given in (3) and, therefore, leads to different iterative algorithms and approximating density functions.

2.4. Mixed model-based penalized splines

Mixed model-based penalized splines are a convenient way to model nonparametric functional relationships in semiparametric regression models, and are amenable to the hierarchical Bayesian structures supported by Infer.NET. The penalized spline of a regression function

f, with mixed model representation, takes the generic form

$$f(x) = \beta_0 + \beta_1 x + \sum_{k=1}^{K} u_k z_k(x), \quad u_k \overset{\text{ind.}}{\sim} N(0, \tau_u^{-1}). \tag{4}$$

Here $z_1(\cdot), \ldots, z_K(\cdot)$ is a set of spline basis functions and τ_u controls the amount of penalization of the spline coefficients $u_1, \ldots, u_k$. Throughout this article, we use *O'Sullivan splines* for the $z_k(\cdot)$. Wand and Ormerod (2008) provides details on their construction. O'Sullivan splines lead to (4) being a low-rank version of smoothing splines, which is also used by the R function `smooth.spline()`.

The simplest semiparametric regression model is the Gaussian response nonparametric regression model

$$y_i \overset{\text{ind.}}{\sim} N(f(x_i), \sigma_\varepsilon^2), \tag{5}$$

where (x_i, y_i), $1 \leq i \leq n$ are pairs of measurements on continuous predictor and response variables. Mixed model-based penalized splines give rise to hierarchical Bayesian models for (5) such as

$$\begin{aligned} y_i|\beta_0, \beta_1, u_1, \ldots, u_K, \tau_\varepsilon &\overset{\text{ind.}}{\sim} N\left(\beta_0 + \beta_1 x_i + \textstyle\sum_{k=1}^{K} u_k z_k(x_i), \tau_\varepsilon^{-1}\right), \\ u_k|\tau_u \overset{\text{ind.}}{\sim} N(0, \tau_u^{-1}), &\quad \beta_0, \beta_1 \overset{\text{ind.}}{\sim} N(0, \tau_\beta^{-1}), \\ \tau_u^{-1/2} \sim \text{Half-Cauchy}(A_u), &\quad \tau_\varepsilon^{-1/2} \sim \text{Half-Cauchy}(A_\varepsilon). \end{aligned} \tag{6}$$

Such models allow nonparametric regression to be performed using Bayesian inference engines such as BUGS and Infer.NET. Our decision to work with precision parameters, rather than the variance parameters (which are common in the semiparametric regression literature), is driven by the former being the standard parametrization in BUGS and Infer.NET.

It is convenient to express (6) using matrix notation. This entails putting

$$\boldsymbol{y} \equiv \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \boldsymbol{X} \equiv \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \boldsymbol{Z} \equiv \begin{bmatrix} z_1(x_1) & \cdots & z_K(x_1) \\ \vdots & \ddots & \vdots \\ z_1(x_n) & \cdots & z_K(x_n) \end{bmatrix} \tag{7}$$

and

$$\boldsymbol{\beta} \equiv \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{u} \equiv \begin{bmatrix} u_1 \\ \vdots \\ u_K \end{bmatrix}. \tag{8}$$

We then re-write (6) as

$$\begin{gathered} \boldsymbol{y} \,|\, \boldsymbol{\beta}, \boldsymbol{u}, \tau_\varepsilon \overset{\text{ind.}}{\sim} N\left(\boldsymbol{X\beta} + \boldsymbol{Zu}, \tau_\varepsilon^{-1}\right), \\ \boldsymbol{u}|\tau_u \sim N(0, \tau_u^{-1}\boldsymbol{I}), \quad \boldsymbol{\beta} \sim N(0, \tau_\beta^{-1}\boldsymbol{I}), \\ \tau_u^{-1/2} \sim \text{Half-Cauchy}(A_u), \quad \tau_\varepsilon^{-1/2} \sim \text{Half-Cauchy}(A_\varepsilon). \end{gathered}$$

The *effective degrees of freedom (edf)* corresponding to (6) is defined to be

$$\text{edf}(\tau_u, \tau_\varepsilon) \equiv \text{tr}\left([\boldsymbol{C}^\top \boldsymbol{C} + \text{blockdiag}\{\tau_\beta^2 \boldsymbol{I}_2, (\tau_u/\tau_\varepsilon)\, \boldsymbol{I}\}]^{-1} \boldsymbol{C}^\top \boldsymbol{C} \right) \tag{9}$$

and is a scale-free measure of the amount of fitting being performed by the penalized splines. Further details on effective degrees of freedom for penalized splines are given in Section 3.13 of Ruppert *et al.* (2003).

Extension to bivariate predictors

The bivariate predictor extension of (5) is

$$y_i \overset{\text{ind.}}{\sim} N(f(\boldsymbol{x}_i), \sigma_\varepsilon^2), \tag{10}$$

where the predictors $\boldsymbol{x}_i$, $1 \le i \le n$, are each 2×1 vectors. In many applications, the $\boldsymbol{x}_i$s correspond to geographical location but they could be measurements on any pair of predictors for which a bivariate mean function might be entertained. Mixed model-based penalized splines can handle this bivariate predictor case by extending (4) to

$$f(\boldsymbol{x}) = \beta_0 + \boldsymbol{\beta}_1^T \boldsymbol{x} + \sum_{k=1}^{K} u_k z_k(\boldsymbol{x}), \quad u_k \overset{\text{ind.}}{\sim} N(0, \tau_u^{-1}) \tag{11}$$

and setting z_k to be appropriate bivariate spline functions. There are several options for doing this (e.g. Ruppert *et al.* 2009, Section 2.2). A relatively simple choice is described here and used in the examples. It is based on thin plate spline theory, and corresponds to Section 13.5 of Ruppert *et al.* (2003). The first step is to choose the number K of bivariate knots and their locations. We denote these 2×1 vectors by $\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_K$. Our default rule for choosing knot locations involves feeding the $\boldsymbol{x}_i$s and K into the clustering algorithm known as CLARA (Kaufman and Rousseeuw 1990) and setting the $\boldsymbol{\kappa}_k$ to be cluster centers. Next, form the matrices

$$\boldsymbol{X} = \begin{bmatrix} 1 & \boldsymbol{x}_1^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_n^T \end{bmatrix},$$

$$\boldsymbol{Z}_K = \begin{bmatrix} \|\boldsymbol{x}_1 - \boldsymbol{\kappa}_1\|^2 \log \|\boldsymbol{x}_1 - \boldsymbol{\kappa}_1\| & \cdots & \|\boldsymbol{x}_1 - \boldsymbol{\kappa}_K\|^2 \log \|\boldsymbol{x}_1 - \boldsymbol{\kappa}_K\| \\ \vdots & \ddots & \vdots \\ \|\boldsymbol{x}_n - \boldsymbol{\kappa}_1\|^2 \log \|\boldsymbol{x}_n - \boldsymbol{\kappa}_1\| & \cdots & \|\boldsymbol{x}_n - \boldsymbol{\kappa}_K\|^2 \log \|\boldsymbol{x}_n - \boldsymbol{\kappa}_K\| \end{bmatrix},$$

$$\text{and} \quad \boldsymbol{\Omega} = \begin{bmatrix} \|\boldsymbol{\kappa}_1 - \boldsymbol{\kappa}_1\|^2 \log \|\boldsymbol{\kappa}_1 - \boldsymbol{\kappa}_1\| & \cdots & \|\boldsymbol{\kappa}_1 - \boldsymbol{\kappa}_K\|^2 \log \|\boldsymbol{\kappa}_1 - \boldsymbol{\kappa}_K\| \\ \vdots & \ddots & \vdots \\ \|\boldsymbol{\kappa}_K - \boldsymbol{\kappa}_1\|^2 \log \|\boldsymbol{\kappa}_K - \boldsymbol{\kappa}_1\| & \cdots & \|\boldsymbol{\kappa}_K - \boldsymbol{\kappa}_K\|^2 \log \|\boldsymbol{\kappa}_K - \boldsymbol{\kappa}_K\| \end{bmatrix}.$$

Based on the singular value decomposition $\boldsymbol{\Omega} = \boldsymbol{U}\text{diag}(\boldsymbol{d})\boldsymbol{V}^T$, compute $\boldsymbol{\Omega}^{1/2} = \boldsymbol{U}\text{diag}(\sqrt{\boldsymbol{d}})\boldsymbol{V}^T$ and then set $\boldsymbol{Z} = \boldsymbol{Z}_K \boldsymbol{\Omega}^{-1/2}$. Then

$$z_k(\boldsymbol{x}_i) = \text{the } (i,k) \text{ entry of } \boldsymbol{Z}$$

with $u_k \overset{\text{ind.}}{\sim} N(0, \tau_u^{-1})$.

3. Simple semiparametric model

The first example involves the simple semiparametric regression model

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \textstyle\sum_{k=1}^{K} u_k z_k(x_{2i}) + \varepsilon_i, \quad \varepsilon_i \overset{\text{ind.}}{\sim} N(0, \tau_\varepsilon^{-1}), \quad 1 \le i \le n, \tag{12}$$

where $z_k(\cdot)$ is a set of spline basis functions as described in Section 2.4. The corresponding Bayesian mixed model can then be represented by

$$\begin{gathered}\boldsymbol{y}|\,\boldsymbol{\beta}, \boldsymbol{u}, \tau_\varepsilon \sim N(\boldsymbol{X\beta} + \boldsymbol{Zu}, \tau_\varepsilon^{-1}\boldsymbol{I}), \quad \boldsymbol{u}|\,\tau_u \sim N(\boldsymbol{0}, \tau_u^{-1}\boldsymbol{I}), \\ \boldsymbol{\beta} \sim N(\boldsymbol{0}, \tau_\beta^{-1}\boldsymbol{I}), \quad \tau_u^{-1/2} \sim \text{Half-Cauchy}(A_u), \quad \tau_\varepsilon^{-1/2} \sim \text{Half-Cauchy}(A_\varepsilon),\end{gathered} \tag{13}$$

where τ_β, A_u and A_ε are user-specified hyperparameters and

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \ \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}, \ \boldsymbol{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_K \end{bmatrix},$$

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & x_{21} \\ \vdots & \vdots & \\ 1 & x_{1n} & x_{2n} \end{bmatrix}, \ \boldsymbol{Z} = \begin{bmatrix} z_1(x_{21}) & \cdots & z_K(x_{21}) \\ \vdots & \ddots & \vdots \\ z_1(x_{2n}) & \cdots & z_K(x_{2n}) \end{bmatrix}.$$

Infer.NET 2.5, Beta 2 does not support direct fitting of model (13) under product restriction

$$q(\boldsymbol{\beta}, \boldsymbol{u}, \tau_u, \tau_\varepsilon) = q(\boldsymbol{\beta}, \boldsymbol{u})\, q(\tau_u, \tau_\varepsilon). \tag{14}$$

We get around this by employing the same trick as that described Section 3.2 of Wang and Wand (2011). It entails the introduction of the auxiliary $n \times 1$ data vector $\boldsymbol{a}$. By setting the observed data for $\boldsymbol{a}$ equal to $\boldsymbol{0}$ and by assuming a very small number for κ, fitting the model in (15) provides essentially the same result as fitting the model in (13). The actual model implemented in Infer.NET is

$$\begin{gathered}\boldsymbol{y}|\boldsymbol{\beta}, \boldsymbol{u}, \tau_\varepsilon \sim N(\boldsymbol{X\beta} + \boldsymbol{Zu}, \tau_\varepsilon^{-1}\boldsymbol{I}), \quad \boldsymbol{a}|\,\boldsymbol{\beta}, \boldsymbol{u}, \tau_u \sim N\left(\begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{u} \end{bmatrix}, \begin{bmatrix} \tau_\beta^{-1}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I} \end{bmatrix}\right), \\ \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{u} \end{bmatrix} \sim N(\boldsymbol{0}, \kappa^{-1}\boldsymbol{I}), \quad \tau_u|\,b_u \sim \text{Gamma}(\tfrac{1}{2}, b_u), \\ b_u \sim \text{Gamma}(\tfrac{1}{2}, 1/A_u^2), \quad \tau_\varepsilon|\,b_\varepsilon \sim \text{Gamma}(\tfrac{1}{2}, b_\varepsilon), \quad b_\varepsilon \sim \text{Gamma}(\tfrac{1}{2}, 1/A_\varepsilon^2),\end{gathered} \tag{15}$$

with the inputted $\boldsymbol{a}$ vector containing zeroes. While the full Infer.NET code is included in the Appendix, we will confine discussion here to the key parts of the code that specify (15).

Specification of the prior distributions for τ_u and τ_ε can be achieved using the following Infer.NET code:

```
Variable<double> tauu = Variable.GammaFromShapeAndRate(0.5,                (16)
   Variable.GammaFromShapeAndRate(0.5,Math.Pow(Au,-2))).Named("tauu");
Variable<double> tauEps = Variable.GammaFromShapeAndRate(0.5,
   Variable.GammaFromShapeAndRate(0.5,Math.Pow(Aeps,-2))).Named("tauEps");
```

while the likelihood in (15) is specified by the following line:

```
y[index] = Variable.GaussianFromMeanAndPrecision(                          (17)
   Variable.InnerProduct(betauWork,cvec[index]),tauEps);
```

The variable `index` represents a range of integers from 1 to n and enables loop-type structures. Finally, the inference engine is specified to be variational message passing via the code:

```
InferenceEngine engine = new InferenceEngine();                            (18)
engine.Algorithm = new VariationalMessagePassing();
engine.NumberOfIterations = nIterVB;
```

with `nIterVB` denoting the number of mean field variational Bayes iterations. Note that this Infer.NET code is treating the coefficient vector

$$\begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{u} \end{bmatrix}$$

as an entity, in keeping with product restriction (14).

Figures 1 and 2 summarize the results from Infer.NET fitting of (15) to a data set on the yields (g/plant) of 84 white Spanish onions crops in two locations: Purnong Landing and Virginia, South Australia. The response variable, y is the logarithm of yield, whilst the predictors are indicator of location being Virginia (x_1) and areal density of the plants (plants/m^2) (x_2). The hyperparameters were set at $\tau_\beta = 10^{-10}$, $A_\varepsilon = 10^5$, $A_u = 10^5$ and $\kappa = 10^{-10}$, while the number of mean field variational Bayes iterations was fixed at 100.

As a benchmark, Bayesian inference via MCMC was performed. To this end, the following BUGS program was used:

```
for(i in 1:n)                                                              (19)
{
  mu[i] <- (beta0 + beta1*x1[i]+ beta2*x2[i] + inprod(u[],Z[i,]))
  y[i] ~ dnorm(mu[i],tauEps)
}
for (k in 1:K)
{
  u[k] ~  dnorm(0,tauu)
}

beta0 ~ dnorm(0,1.0E-10) ; beta1 ~ dnorm(0,1.0E-10)
beta2 ~ dnorm(0,1.0E-10) ;
bu ~ dgamma(0.5,1.0E-10) ; tauu ~ dgamma(0.5,bu)
bEps ~ dgamma(0.5,1.0E-10) ; tauEps ~ dgamma(0.5,bEps)
```

A burn-in length of 50000 was used, while 1 million samples were obtained from the posterior distributions. Finally, a thinning factor of 5 was used. The posterior densities for MCMC were produced based on kernel density estimation with plug-in bandwidth selection via the R package **KernSmooth** (Wand and Ripley 2010). Figure 1 displays the fitted regression lines and pointwise 95% credible intervals. The estimated regression lines and credible intervals from Infer.NET and MCMC fitting are highly similar. Figure 2 visualizes the approximate posterior density functions for β_1, τ_ε^{-1} and the effective degrees of freedom of the fit. The variational Bayes approximations for β_1 and τ_ε^{-1} are rather accurate. The approximate posterior density function for the effective degrees of freedom for variational Bayesian inference was obtained based on Monte Carlo samples of size 1 million from the approximate posterior distributions of τ_ε^{-1} and τ_u^{-1}.

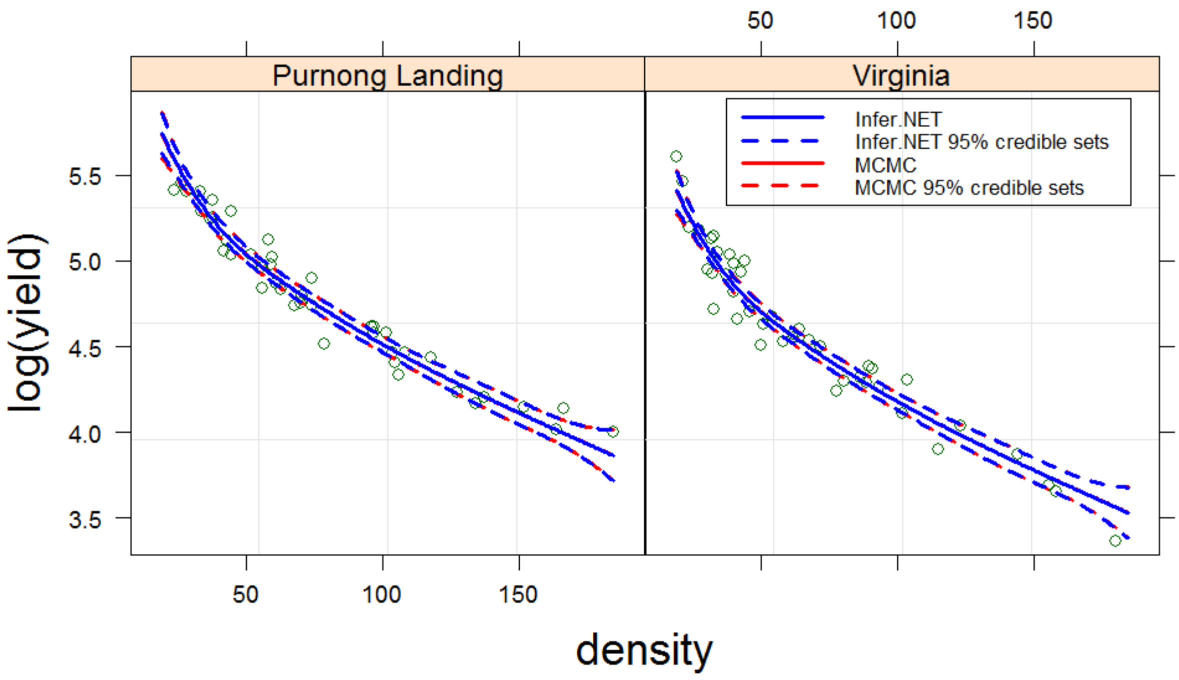


Figure 1: Onions data. Fitted regression line and pointwise 95% credible intervals for variational Bayesian inference by Infer.NET and MCMC.

4. Generalized additive model

The next example illustrates binary response variable regression through the model

$$
\begin{aligned}
&y_i|\boldsymbol{\beta},\boldsymbol{u} \overset{\text{ind.}}{\sim} \text{Bernoulli}(F(\{\boldsymbol{X\beta}+\boldsymbol{Zu}\}_i)), \quad \boldsymbol{u}|\,\tau_u \sim N(\mathbf{0},\tau_u^{-1}\boldsymbol{I}),\\
&\boldsymbol{\beta} \sim N(\mathbf{0},\tau_\beta^{-1}\boldsymbol{I}), \quad \tau_u^{-1/2} \sim \text{Half-Cauchy}(A_u), \quad 1 \le i \le n,
\end{aligned}
\tag{20}
$$

where $F(\cdot)$ denotes an inverse link function and $\boldsymbol{X}$, $\boldsymbol{Z}$, τ_β and A_u are defined as in the previous section. Typical choices of $F(\cdot)$ correspond to logistic regression and probit regression.

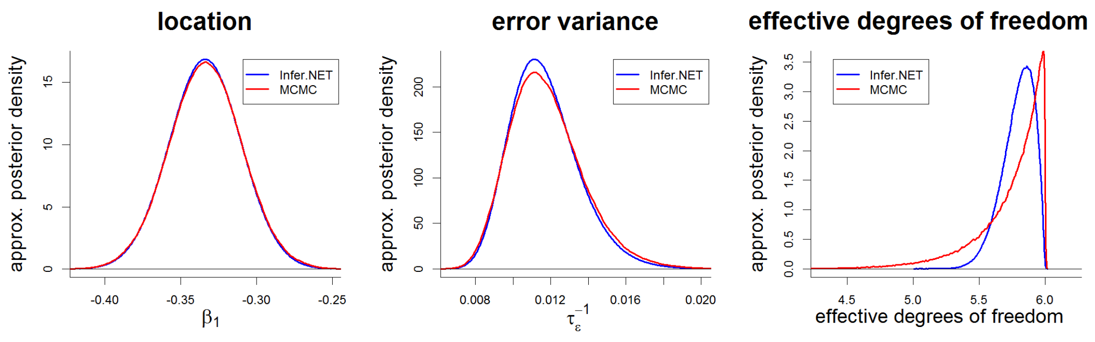


Figure 2: Variational Bayes approximate posterior density functions produced by Infer.NET and MCMC posterior density functions of key model parameters for fitting a simple semi-parametric model to the onions data set.

As before, introduction of the auxiliary variables $\boldsymbol{a} \equiv 0$ enables **Infer.NET** fitting of the Bayesian mixed model

$$
\begin{aligned}
&y_i|\boldsymbol{\beta},\boldsymbol{u} \overset{\text{ind.}}{\sim} \text{Bernoulli}(F(\{\boldsymbol{X\beta}+\boldsymbol{Zu}\}_i)),\\
&\boldsymbol{a}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_u \sim N\left(\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix},\begin{bmatrix}\tau_\beta^{-1}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I}\end{bmatrix}\right),\\
&\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix} \sim N(\boldsymbol{0},\kappa^{-1}\boldsymbol{I}), \quad \tau_u|\,b_u \sim \text{Gamma}(\tfrac{1}{2},b_u),\\
&b_u \sim \text{Gamma}(\tfrac{1}{2},1/A_u^2), \quad 1 \le i \le n.
\end{aligned}
\tag{21}
$$

The likelihood for the logistic regression case is specified as follows

```
VariableArray<bool> y = Variable.Array<bool>(index).Named("y");          (22)
y[index] = Variable.BernoulliFromLogOdds(
   Variable.InnerProduct(betauWork, cvec[index]));
```

while variational message passing is used for fitting purposes. Setting up the prior for τ_u and specifying the inference engine is done as in code chunk (16) and (18), respectively. For probit regression, the last line of (22) is replaced by

```
y[index] = Variable.IsPositive(Variable.GaussianFromMeanAndVariance(      (23)
   Variable.InnerProduct(betauWork, cvec[index]), 1));
```

and expectation propagation is used

```
engine.Algorithm = new ExpectationPropagation();                          (24)
```

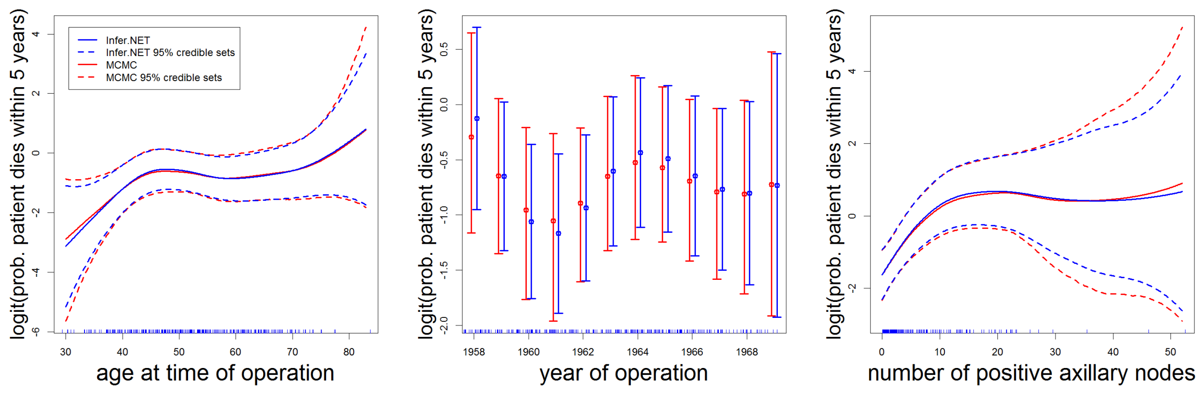
Instead of the half-Cauchy prior in (20), a gamma prior with shape and rate equal to 2 is used for τ_u in the probit regression model

```
Variable<double> tauU = Variable.GammaFromShapeAndRate(2,2);              (25)
```

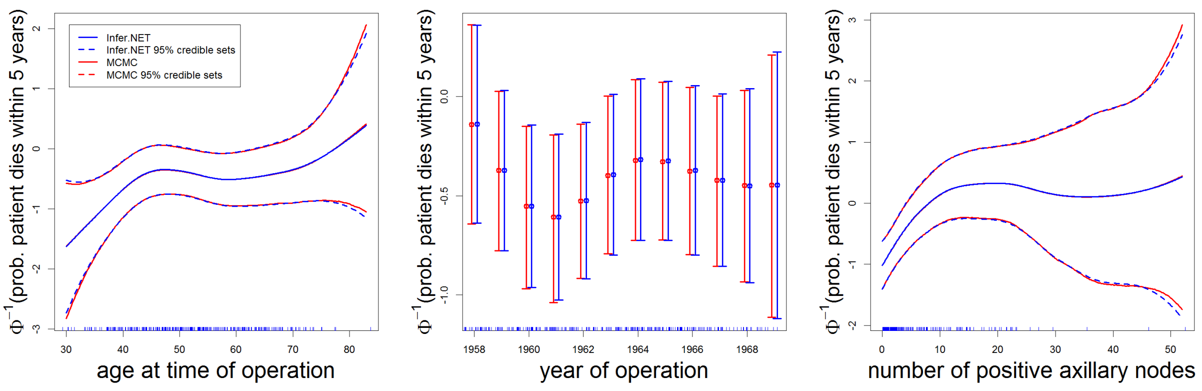
Figure 3 visualizes the results for **Infer.NET** and MCMC fitting of a straightforward extension of model (21) with inverse-logit and probit link functions to a breast cancer data set (Haberman 1976). This data set contains 306 cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. The binary response variable represents whether the patient died within 5 years after operation (`died`), while 3 predictor variables are used: age of patient at the time of operation (`age`), year of operation (`year`) and number of positive axillary nodes (`nodes`) detected. The actual model is

$$\texttt{died}_i|\boldsymbol{\beta},\boldsymbol{u} \overset{\text{ind.}}{\sim} \text{Bernoulli}(F(\beta_0 + f_1(\texttt{age}_i) + f_2(\texttt{year}_i) + f_3(\texttt{nodes}_i))), \quad 1 \le i \le 306.$$

Note that all predictor variables were standardized prior to model fitting and the following hyperparameters were used: $\tau_\beta = 10^{-10}$, $A_u = 10^5$, $\kappa = 10^{-10}$ and the number of variational Bayes iterations was 100. Figure 3 illustrates that there is good agreement between the results from **Infer.NET** and MCMC.



(a)



(b)

Figure 3: Breast cancer data. Fitted regression lines and pointwise 95% credible intervals for logit (a) and probit (b) regression via variational Bayesian inference by **Infer.NET** and MCMC.

5. Robust nonparametric regression based on the t-distribution

A popular model-based approach for robust regression is to model the response variable to have a t-distribution. Outliers occur with moderate probability for low values of the t-distribution's degrees of freedom parameter (Lange *et al.* 1989). More recently, Staudenmayer *et al.* (2009) proposed a penalized spline mixed model approach to nonparametric regression using the t-distribution.

The robust nonparametric regression model that we consider here is a Bayesian variant of that treated by Staudenmayer *et al.* (2009):

$$
\begin{gathered}
y_i|\,\beta_0,\beta_1,\boldsymbol{u},\tau_\varepsilon,\nu \overset{\text{ind.}}{\sim} t\left(\beta_0+\beta_1\,x_i+\textstyle\sum_{k=1}^K u_k\,z_k(x_i),\tau_\varepsilon^{-1},\nu\right),\quad 1\le i\le n,\\
\boldsymbol{u}|\,\tau_u\sim N(\boldsymbol{0},\tau_u^{-1}\boldsymbol{I}),\quad \boldsymbol{\beta}\sim N(\boldsymbol{0},\tau_\beta^{-1}\boldsymbol{I}),\\
\tau_u^{-1/2}\sim \text{Half-Cauchy}(A_u),\quad \tau_\varepsilon^{-1/2}\sim\text{Half-Cauchy}(A_\varepsilon),\\
p(\nu)\ \text{discrete on a finite set } N.
\end{gathered}
\tag{26}
$$

Restricting the prior distribution of ν to be that of a discrete random variable allows Infer.NET fitting of the model using *structured* mean field variational Bayes (Saul and Jordan 1996). This extension of ordinary mean field variational Bayes is described in Section 3.1 of Wand *et al.* (2011). Since variational message passing is a special case of mean field variational Bayes this extension also applies. Model (26) can be fitted through calls to Infer.NET with ν fixed at each value in N. The results of each of these fits are combined afterwards. Details are given below.

Another challenge concerning (26) is that Infer.NET does not support t-distribution specifications. We get around this by appealing to the result

$$
\text{if}\quad x|\,g\sim\ N\left(\mu,(g\tau)^{-1}\right)\quad\text{and}\quad g\sim\text{Gamma}(\tfrac{\nu}{2},\tfrac{\nu}{2})\quad\text{then}\quad x\sim t(\mu,\tau^{-1},\nu).
\tag{27}
$$

As before, we use the $\boldsymbol{a}\equiv\boldsymbol{0}$ auxiliary data trick described in Section 3 and the Half-Cauchy representation (1). These lead to following suite of models, for each fixed $\nu\in N$, needing to be run in Infer.NET:

$$
\begin{gathered}
\boldsymbol{y}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_\varepsilon\sim N(\boldsymbol{X\beta}+\boldsymbol{Zu},\tau_\varepsilon^{-1}\text{diag}(1/\boldsymbol{g})),\\
\boldsymbol{a}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_u\sim N\left(\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix},\begin{bmatrix}\tau_\beta^{-1}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I}\end{bmatrix}\right),\qquad \begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix}\sim N(\boldsymbol{0},\kappa^{-1}\boldsymbol{I})\\
\boldsymbol{u}|\,\tau_u\sim N(\boldsymbol{0},\tau_u^{-1}\boldsymbol{I}),\quad g_i|\,\nu\overset{\text{ind.}}{\sim}\text{Gamma}(\tfrac{\nu}{2},\tfrac{\nu}{2}),\quad \boldsymbol{\beta}\sim N(\boldsymbol{0},\tau_\beta^{-1}\boldsymbol{I}),\\
\tau_u|\,b_u\sim\text{Gamma}(\tfrac12,b_u),\quad b_u\sim\text{Gamma}(\tfrac12,1/A_u^2),\\
\tau_\varepsilon|\,b_\varepsilon\sim\text{Gamma}(\tfrac12,b_\varepsilon),\quad b_\varepsilon\sim\text{Gamma}(\tfrac12,1/A_\varepsilon^2),
\end{gathered}
\tag{28}
$$

where the matrix notation of (7), (8) is being used, $\boldsymbol{g}\equiv[g_1,\ldots,g_n]^T$ and τ_β, A_u and A_ε are user-specified hyperparameters with default values $\tau_\beta=10^{-10}$, $A_u=A_\varepsilon=10^5$. The prior for ν was set to be a uniform distribution over the atom set N, set to be 30 equally-spaced numbers between 0.05 and 10 inclusive.

After obtaining fits of (28) for each $\nu \in N$, the approximate posterior densities are obtained from

$$q(\boldsymbol{\beta}, \boldsymbol{u}) = \sum_{\nu \in N} q_\nu(\nu) q(\boldsymbol{\beta}, \boldsymbol{u}|\nu), \quad q(\tau_u) = \sum_{\nu \in N} q_\nu(\nu)\, q(\tau_u|\nu),$$
$$q(\tau_\varepsilon) = \sum_{\nu \in N} q(\nu)\, q(\tau_\varepsilon|\nu), \quad \text{with} \quad q(\nu) = \underline{p}(\nu|\boldsymbol{y}) = \frac{p(\nu)\underline{p}(\boldsymbol{y}|\nu)}{\sum_{\nu' \in N} p(\nu')\underline{p}(\boldsymbol{y}|\nu')},$$

and $\underline{p}(\boldsymbol{y}|\nu)$ denotes the variational lower bound on the conditional likelihood $p(\boldsymbol{y}|\nu)$.

Specification of the prior distribution for $\boldsymbol{g}$ is achieved using the following Infer.NET code:

```
VariableArray<double> g = Variable.Array<double>(index);                (29)
g[index] = Variable.GammaFromShapeAndRate(nu/2,2/nu).ForEach(index);
```

while the likelihood in (28) is specified by:

```
y[index] = Variable.GaussianFromMeanAndPrecision(                       (30)
   Variable.InnerProduct(betauWork,cvec[index]),g[index]*tauEps);
```

The lower bound $\log \underline{p}(\boldsymbol{y}|\nu)$ can be obtained by creating a mixture of the current model with an empty model in Infer.NET. The learned mixing weight is then equal to the marginal log-likelihood. Therefore, an auxiliary Bernoulli variable is set up:

```
Variable<bool> auxML = Variable.Bernoulli(0.5).Named("auxML");          (31)
IfBlock model = Variable.If(auxML);
```

The normal code for fitting the model in (28) is then enclosed with

```
IfBlock model = Variable.If(auxML);                                     (32)
```

and

```
model.CloseBlock();                                                     (33)
```

Finally, the lower bound $\log \underline{p}(\boldsymbol{y}|\nu)$ is obtained from:

```
double marginalLogLikelihood = engine.Infer<Bernoulli>(auxML).LogOdds;  (34)
```

Figure 4 presents the results of the structured mean field variational Bayes analysis using Infer.NET fitting of model (28) to a data set on a respiratory experiment conducted by Professor Russ Hauser at Harvard School of Public Health, Boston, USA. The data correspond to 60 measurements on one subject during two separate respiratory experiments. The response variable y_i represents the log of the adjusted time of exhalation for x_i equal to the time in seconds since exposure to air containing particulate matter. The adjusted time of exhalation is obtained by subtracting the average time of exhalation at baseline, prior to exposure to filtered air. Interest centers upon the mean response as a function of time. The predictor and response variable were both standardized prior to Infer.NET analysis. The following hyper-parameters were chosen: $\tau_\beta = 10^{-10}$, $A_\varepsilon = 10^5$, $A_u = 10^5$ and $\kappa = 10^{-10}$, while the number of variational Bayes iterations equaled 100. Bayesian inference via MCMC was performed as a benchmark.

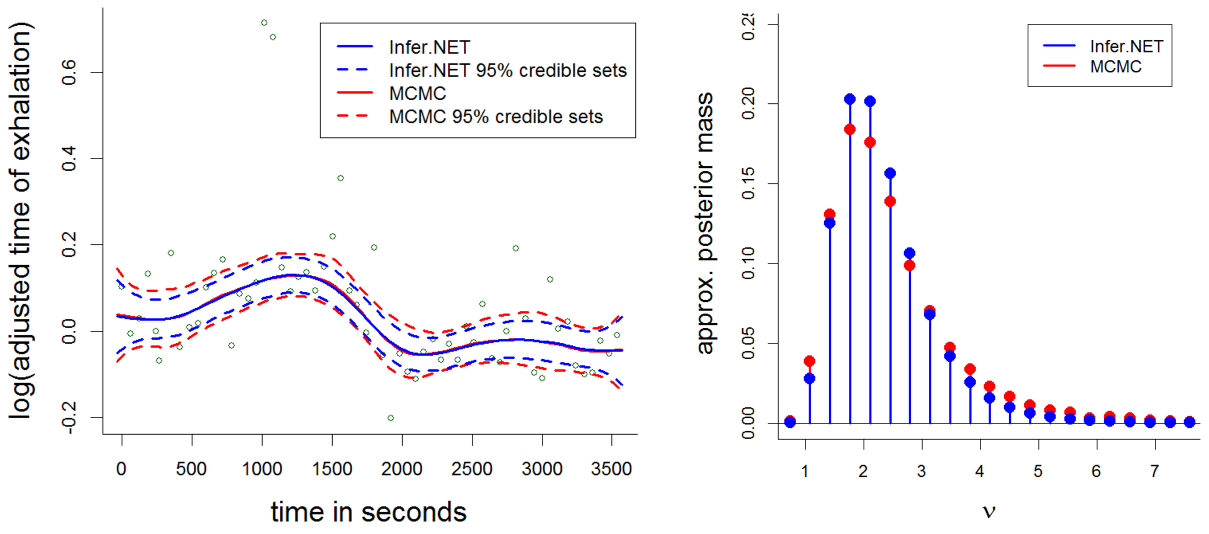


Figure 4: Structured mean field variational Bayesian, based on Infer.NET, and MCMC fitting of the robust nonparametric regression model (26) to the respiratory experiment data. Left: Posterior mean and pointwise 95% credible sets for the regression function. Right: Approximate posterior function for the degrees of freedom parameter ν.

Figure 4 shows that the variational Bayes fit and pointwise 95% credible sets are close to the ones obtained using MCMC. Finally, the approximate posterior probability function and the posterior from MCMC for the degrees of freedom ν are compared. The Infer.NET and MCMC results coincide quite closely.

6. Semiparametric mixed model

Since semiparametric regression models based on penalized splines fit in the mixed model framework, semiparametric longitudinal data analysis can be performed by fusion with classical mixed models (Ruppert *et al.* 2003). In this section we illustrate the use of Infer.NET for fitting the class of semiparametric mixed models having the form:

$$y_{ij}|\boldsymbol{\beta},\boldsymbol{u},U_i,\tau_\varepsilon \overset{\text{ind.}}{\sim} N\left(\beta_0+\boldsymbol{\beta}^T\boldsymbol{x}_{ij}+U_i+\sum_{k=1}^{K}u_k\,z_k(s_{ij}),\tau_\varepsilon^{-1}\right),$$
$$U_i|\,\tau_U \overset{\text{ind.}}{\sim} N(0,\tau_U^{-1}),\quad 1\le j\le n_i,\quad 1\le i\le m, \tag{35}$$

for analysis of the longitudinal data-sets such as that described in Bachrach *et al.* (1999). Here $\boldsymbol{x}_{ij}$ is a vector of predictors that enter the model linearly and s_{ij} is another predictor that enters the model non-linearly via penalized splines. For each $1\le i\le m$, U_i denotes the random intercept for the ith subject. The corresponding Bayesian mixed model is represented by

$$\begin{gathered}
\boldsymbol{y}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_\varepsilon \sim N(\boldsymbol{X\beta}+\boldsymbol{Zu},\tau_\varepsilon^{-1}\boldsymbol{I}), \quad \boldsymbol{u}|\,\tau_U,\tau_u \sim N\left(\boldsymbol{0},\begin{bmatrix}\tau_U^{-1}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I}\end{bmatrix}\right),\\
\boldsymbol{\beta}\sim N(\boldsymbol{0},\tau_\beta^{-1}\boldsymbol{I}), \quad \tau_u^{-1/2}\sim \text{Half-Cauchy}(A_u), \quad \tau_\varepsilon^{-1/2}\sim\text{Half-Cauchy}(A_\varepsilon),\\
\tau_U^{-1/2}\sim\text{Half-Cauchy}(A_U),
\end{gathered} \tag{36}$$

where $\boldsymbol{y},\boldsymbol{\beta}$ and $\boldsymbol{X}$ are defined in a similar manner as in the previous sections and τ_β, A_u, A_ε and A_U are user-specified hyperparameters. Introduction of the random intercepts results in

$$\boldsymbol{u}=\begin{bmatrix}U_1\\ \vdots\\ U_m\\ u_1\\ \vdots\\ u_K\end{bmatrix}, \quad \boldsymbol{Z}=\begin{bmatrix}1 & \cdots & 0 & z_1(s_{11}) & \cdots & z_K(s_{11})\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ 1 & \cdots & 0 & z_1(s_{1n_1}) & \cdots & z_K(s_{1n_1})\\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & \cdots & 1 & z_1(s_{m1}) & \cdots & z_K(s_{m1})\\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots\\ 0 & \cdots & 1 & z_1(s_{mn_m}) & \cdots & z_K(s_{mn_m})\end{bmatrix}.$$

We again use the auxiliary data vector $\boldsymbol{a}\equiv\boldsymbol{0}$ to allow direct Infer.NET fitting of the Bayesian longitudinal penalized spline model

$$\begin{gathered}
\boldsymbol{y}|\boldsymbol{\beta},\boldsymbol{u},\tau_\varepsilon\sim N(\boldsymbol{X\beta}+\boldsymbol{Zu},\tau_\varepsilon^{-1}\boldsymbol{I}), \quad \begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix}\sim N(\boldsymbol{0},\kappa^{-1}\boldsymbol{I}),\\
\boldsymbol{a}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_U,\tau_u\sim N\left(\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix},\begin{bmatrix}\tau_\beta^{-1}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \tau_U^{-1}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \tau_u^{-1}\boldsymbol{I}\end{bmatrix}\right),\\
\tau_u|\,b_u\sim\text{Gamma}(\tfrac{1}{2},b_u), \quad b_u\sim\text{Gamma}(\tfrac{1}{2},1/A_u^2), \quad \tau_\varepsilon|\,b_\varepsilon\sim\text{Gamma}(\tfrac{1}{2},1/b_\varepsilon),\\
b_\varepsilon\sim\text{Gamma}(\tfrac{1}{2},1/A_\varepsilon^2), \quad \tau_U|\,b_U\sim\text{Gamma}(\tfrac{1}{2},b_U), \quad b_U\sim\text{Gamma}(\tfrac{1}{2},1/A_U^2).
\end{gathered} \tag{37}$$

Specification of the prior distribution for τ_U can be achieved in Infer.NET in a similar manner as prior specification in code chunk (16).

Figure 5 shows the Infer.NET fits of (37) to the spinal bone mineral density data (Bachrach *et al.* 1999). A population of 230 female subjects aged between 8 and 27 was followed over time and each subject contributed either one, two, three, or four spinal bone mineral density measurements. Age enters the model non-linearly and corresponds to s_{ij} in (35). Data on ethnicity are available and the entries of $\boldsymbol{x}_{ij}$ correspond to the indicator variables for Black (x_{1ij}), Hispanic (x_{2ij}) and White (x_{3ij}), with Asian ethnicity corresponding to the baseline. The following hyperparameters were chosen: $\tau_\beta = 10^{-10}$, $A_\varepsilon = 10^5$, $A_u = 10^5$, $A_U = 10^5$ and $\kappa = 10^{-10}$, while the number of mean field variational Bayes iterations was set to 100. Figure 5 suggests that there exists a statistically significant difference in mean spinal bone

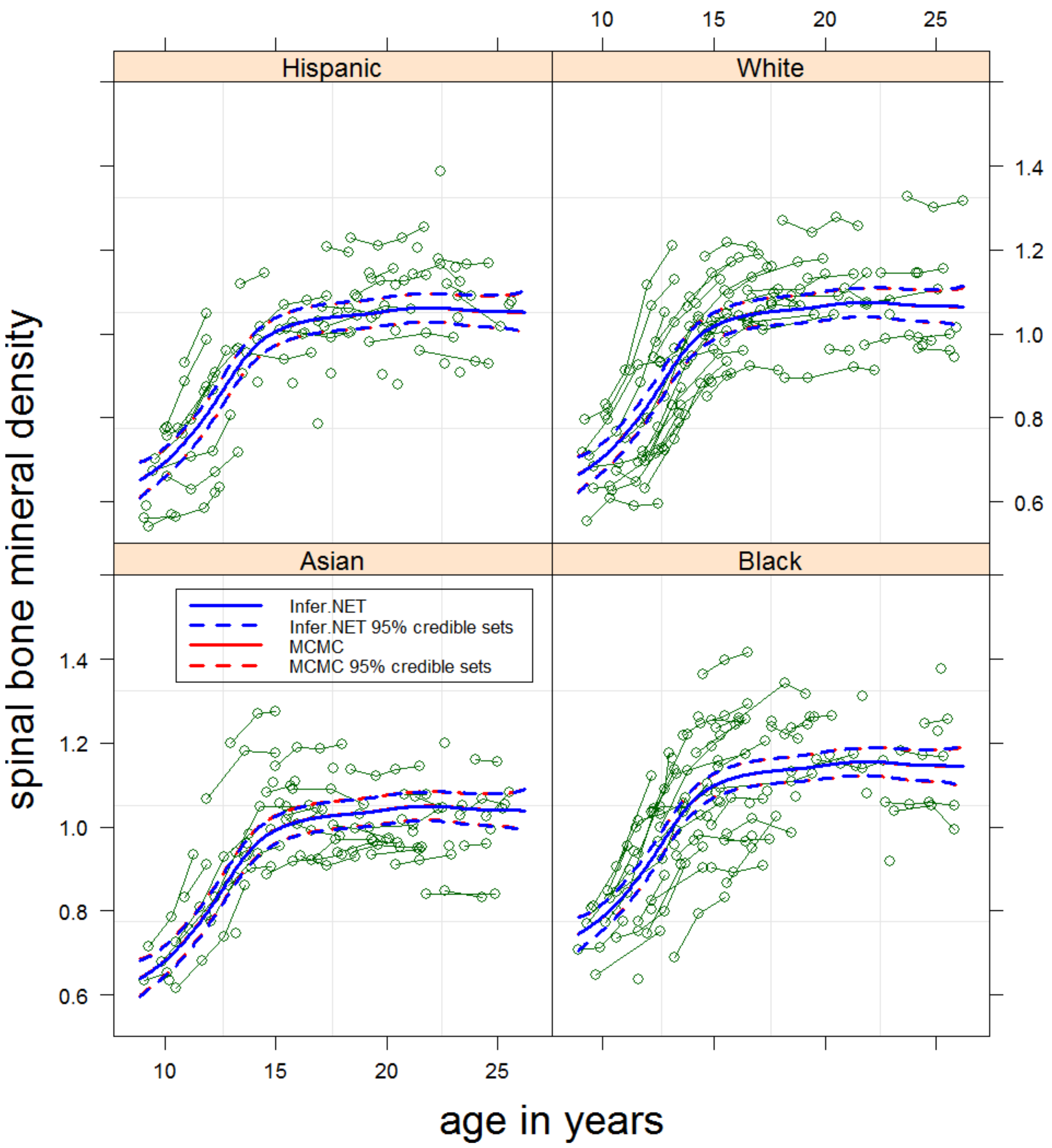


Figure 5: Spinal bone mineral density data. Fitted regression line and pointwise 95% credible intervals based on mean field variational Bayesian inference by Infer.NET and MCMC.

mineral density between Asian and Black subjects. This difference is confirmed by the approximate posterior density functions in Figure 6. No statistically significant difference is found between Asian and Hispanic subjects and between Asian and White subjects.

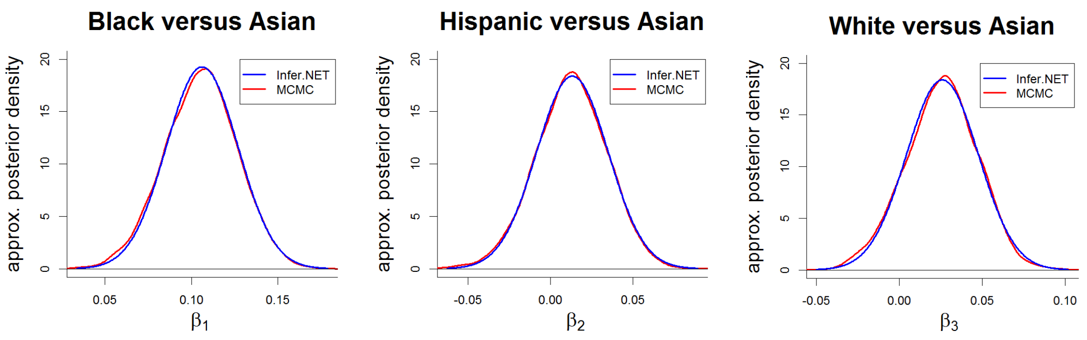


Figure 6: Variational Bayes approximate posterior density functions produced by Infer.NET of ethnic group parameters for fitting a simple semiparametric mixed model to the spinal bone mineral density data set.

7. Geoadditive model

We turn our attention to geostatistical data, for which the response variable is geographically referenced and the extension of generalized additive models, known as *geoadditive models* (Kammann and Wand 2003), applies. Such models allow for a pair of continuous predictors, typically geographical location, to have a bivariate functional impact on the mean response. An example geoadditive model is

$$y_i \sim N\left(f_1(x_{1i}) + f_2(x_{2i}) + f_{\text{geo}}(x_{3i}, x_{4i}), \tau_\varepsilon^{-1}\right), \quad 1 \le i \le n. \tag{38}$$

It can be handled using the spline basis described in Section 2.4 as follows:

$$\begin{aligned} y_i |\, \boldsymbol{\beta}, \boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}^{\text{geo}}, \tau_\varepsilon \overset{\text{ind.}}{\sim} N\Big(&\beta_0 + \beta_1\, x_{1i} + \beta_2\, x_{2i} + \beta_3\, x_{3i} + \beta_4\, x_{4i} \\ &+ \sum_{k=1}^{K_1} u_{1k}\, z_{1k}(x_{1i}) + \sum_{k=1}^{K_2} u_{2k}\, z_{2k}(x_{2i}) + \sum_{k=1}^{K_{\text{geo}}} u_k^{\text{geo}}\, z_k^{\text{geo}}(x_{3i}, x_{4i}), \tau_\varepsilon^{-1}\Big), \quad 1 \le i \le n, \end{aligned} \tag{39}$$

where the z_{1k} and z_{2k} are univariate spline basis functions as used in each of this article's previous examples and the z_k^{geo} are the bivariate spline basis functions, described in Section 2.4. The corresponding Bayesian mixed model is represented by

$$\begin{gathered} \boldsymbol{y} |\, \boldsymbol{\beta}, \boldsymbol{u}, \tau_\varepsilon \sim N(\boldsymbol{X\beta} + \boldsymbol{Zu}, \tau_\varepsilon^{-1}\boldsymbol{I}), \quad \boldsymbol{\beta} \sim N(\boldsymbol{0}, \tau_\beta^{-1}\boldsymbol{I}), \\ \boldsymbol{u} \equiv \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}^{\text{geo}} \end{bmatrix} \Bigg| \tau_{u1}, \tau_{u2}, \tau_{\text{geo}} \sim N\left(\boldsymbol{0}, \begin{bmatrix} \tau_{u1}^{-1}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \tau_{u2}^{-1}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \tau_{\text{geo}}^{-1}\boldsymbol{I} \end{bmatrix}\right), \\ \tau_u^{-1/2} \sim \text{Half-Cauchy}(A_u), \quad \tau_\varepsilon^{-1/2} \sim \text{Half-Cauchy}(A_\varepsilon), \\ \tau_{\text{geo}}^{-1/2} \sim \text{Half-Cauchy}(A_{\text{geo}}), \end{gathered} \tag{40}$$

where $\boldsymbol{y}, \boldsymbol{\beta}$, $\boldsymbol{u}$ and $\boldsymbol{X}$ are defined in a similar manner as in the previous sections and τ_β, A_u,

A_ε and A_{geo} are user-specified hyperparameters. The $\boldsymbol{Z}$ matrix is

$$\boldsymbol{Z} = \begin{bmatrix} z_{11}(x_{11}) & \cdots & z_{1K_1}(x_{11}) & z_{21}(x_{21}) & \cdots & z_{2K_2}(x_{21}) & z_1^{\text{geo}}(x_{31},x_{41}) & \cdots & z_{K_{\text{geo}}}^{\text{geo}}(x_{31},x_{41}) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ z_{11}(x_{1n}) & \cdots & z_{1K_1}(x_{1n}) & z_{21}(x_{2n}) & \cdots & z_{2K_2}(x_{2n}) & z_1^{\text{geo}}(x_{3n},x_{4n}) & \cdots & z_{K_{\text{geo}}}^{\text{geo}}(x_{3n},x_{4n}) \end{bmatrix}.$$

Infer.NET can be used to fit the Bayesian geoadditive model

$$\boldsymbol{y}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_\varepsilon \sim N(\boldsymbol{X\beta}+\boldsymbol{Zu},\tau_\varepsilon^{-1}\boldsymbol{I}), \quad \begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix} \sim N(\boldsymbol{0},\kappa^{-1}\boldsymbol{I}),$$

$$\boldsymbol{a}|\,\boldsymbol{\beta},\boldsymbol{u},\tau_u,\tau_{\text{geo}} \sim N\left(\begin{bmatrix}\boldsymbol{\beta}\\ \boldsymbol{u}\end{bmatrix}, \begin{bmatrix} \tau_\beta^{-1}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \tau_{u1}^{-1}\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \tau_{u2}^{-1}\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \tau_{\text{geo}}^{-1}\boldsymbol{I} \end{bmatrix}\right),$$

$$\tau_{u1}|\,b_{u1} \sim \text{Gamma}(\tfrac{1}{2},b_{u1}), \quad b_{u1} \sim \text{Gamma}(\tfrac{1}{2},1/A_{u1}^2), \quad \tau_{u2}|\,b_{u2} \sim \text{Gamma}(\tfrac{1}{2},b_{u2}),$$

$$b_{u2} \sim \text{Gamma}(\tfrac{1}{2},1/A_{u2}^2), \quad \tau_\varepsilon|\,b_\varepsilon \sim \text{Gamma}(\tfrac{1}{2},b_\varepsilon), b_\varepsilon \sim \text{Gamma}(\tfrac{1}{2},1/1A_\varepsilon^2),$$

$$\tau_{\text{geo}}|\,b_{\text{geo}} \sim \text{Gamma}(\tfrac{1}{2},b_{\text{geo}}), \quad b_{\text{geo}} \sim \text{Gamma}(\tfrac{1}{2},1/A_{\text{geo}}^2)$$

where $\boldsymbol{a} \equiv \boldsymbol{0}$ as in all previous examples.

We illustrate geoadditive model fitting in Infer.NET using data on residential property prices of 37,676 residential properties that were sold in Sydney, Australia, during 2001. The data were assembled as part of of an unpublished study by A. Chernih and M. Sherris at the University of New South Wales, Australia. The response variable is the logarithm of sale price in Australian dollars. Apart from geographical location, several predictor variables are available. For this example we selected weekly income in Australian dollars, the distance from the coastline in kilometres, the particulate matter 10 level and the nitrogen dioxide level. The model is a straightforward extension of the geoadditive model conveyed by (38)–(40). In addition, all predictor variables and the dependent variable were standardized before model fitting. The hyperparameters set to $\tau_\beta = 10^{-10}$, $A_\varepsilon = 10^5 = A_{u_1} = \ldots = A_{u_4} = A_{\text{geo}} = 10^5$, while the number of variational message passing iterations was equal to 100 and κ set to 10^{-10}.

Figure 7 summarizes the housing prices for the Sydney metropolitan area based on the fitted Infer.NET model, while fixing the four predictor variables at their mean levels. This result clearly shows that the sale price is higher for houses in Sydney's eastern and northern suburbs whereas it is strongly decreased for houses in Sydney's western suburbs.

Figure 8 visualizes the fitted regression line and pointwise 95% credible intervals for income, distance from the coastline, particulate matter 10 level and nitrogen dioxide level at a fixed geographical location (longitude = 151.10°, latitude = −33.91°). As expected, a higher income is associated with a higher sale price and houses close to the coastline are more expensive. The sale price is lower for a higher particulate matter 10 level, while a lower nitrogen dioxide level is generally associated with a lower sale price.

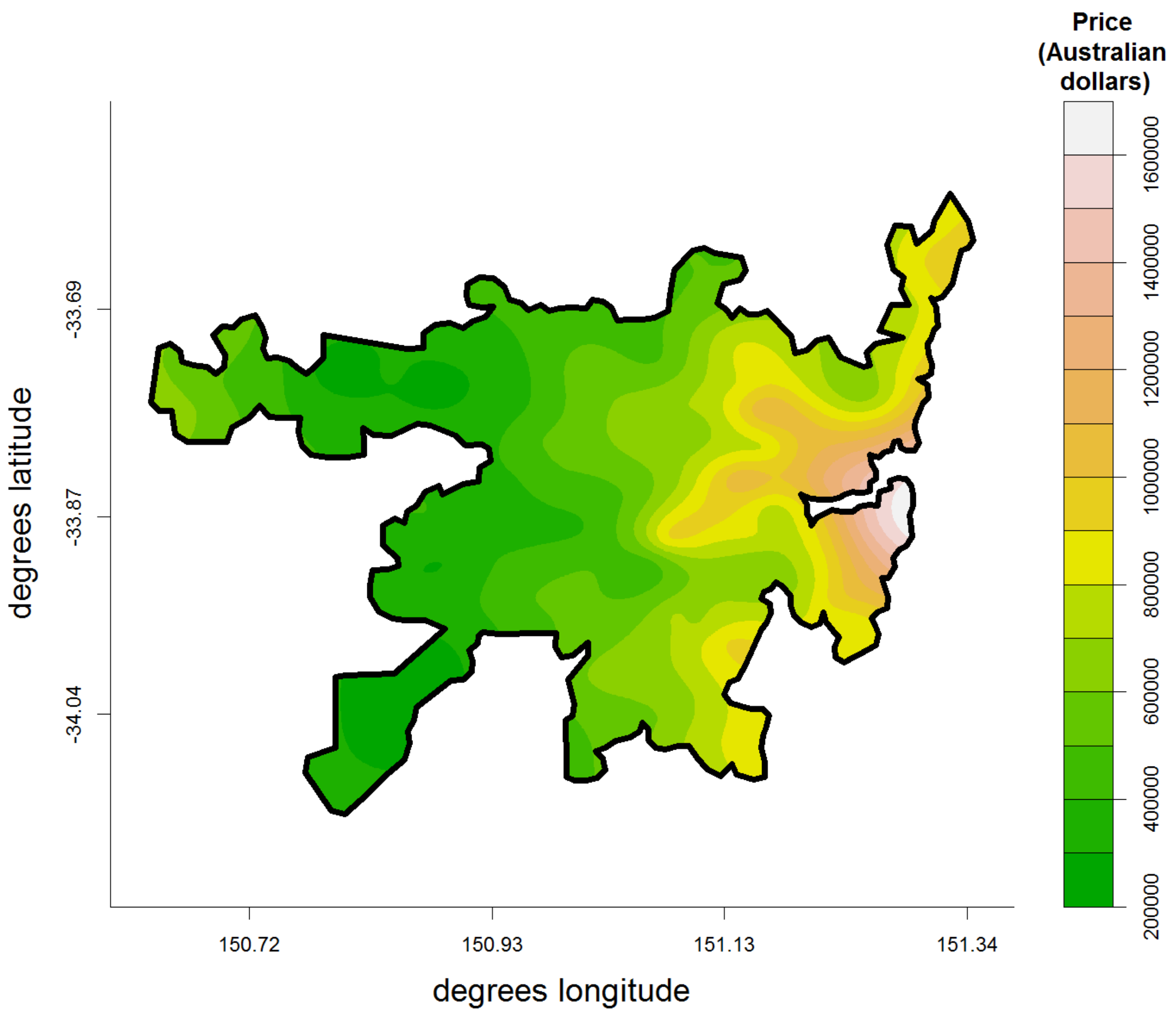


Figure 7: Sydney real estate data. Estimated housing prices for the Sydney metropolitan area for variational Bayesian inference by Infer.NET.

8. Bayesian lasso regression

A Bayesian approach to lasso (least absolute shrinkage selection operator) regression was proposed by Park and Casella (2008). For linear regression, the lasso approach essentially involves the replacing

$$\beta_j\,|\,\tau_\beta \overset{\text{ind.}}{\sim} N(0,\tau_\beta^{-1}) \quad \text{by} \quad \beta_j\,|\,\tau_\beta \overset{\text{ind.}}{\sim} \text{Laplace}(0,\tau_\beta^{-1}) \tag{41}$$

where β_j is the coefficient of the jth predictor variable. Replacement (41) corresponds to the form of the penalty changing from

$$\lambda \sum_{j=1}^{p} \beta_j^2 \quad \text{to} \quad \lambda \sum_{j=1}^{p} |\beta_j|.$$

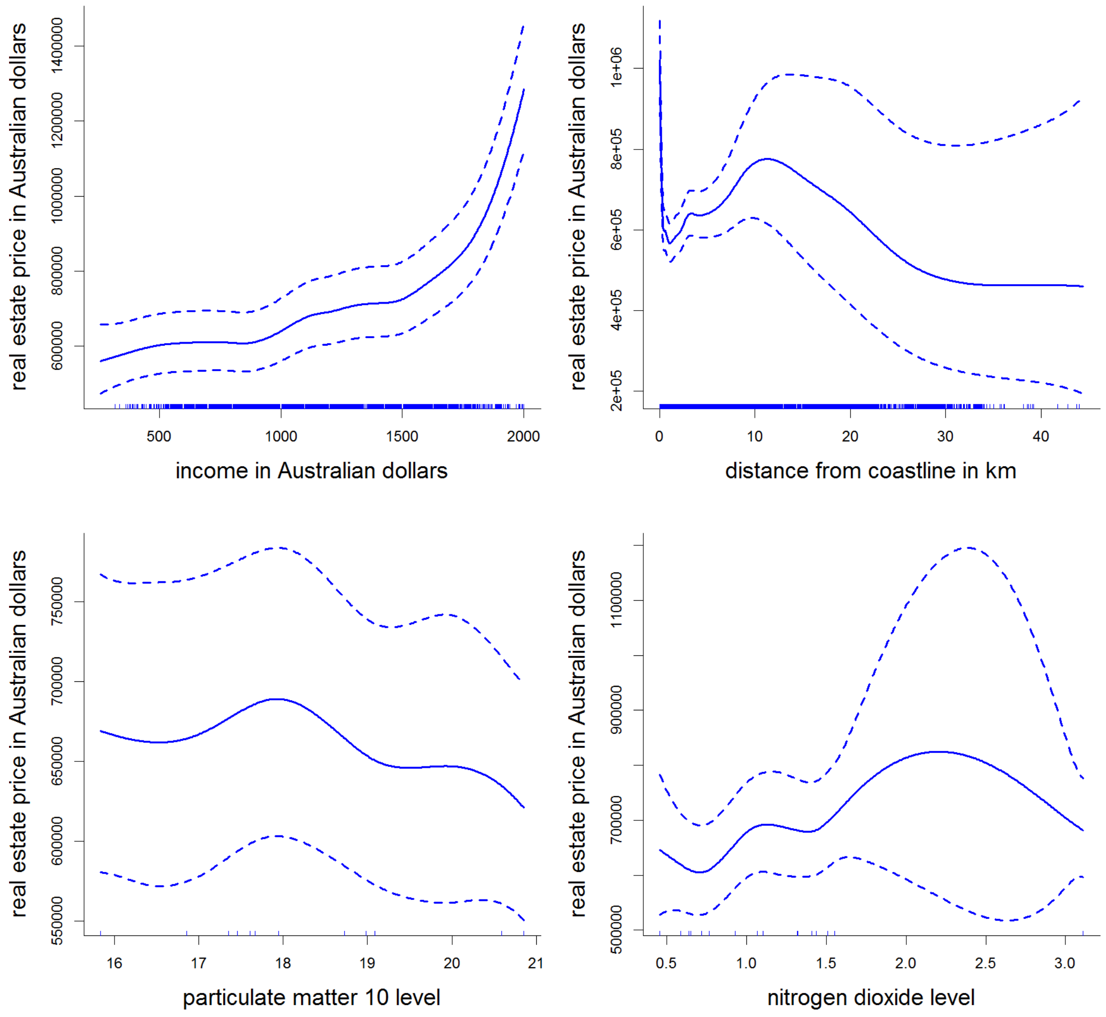


Figure 8: Sydney real estate data. Fitted regression line and pointwise 95% credible intervals for variational Bayesian inference by Infer.NET.

For frequentist lasso regression (Tibshirani 1996) the latter penalty produces sparse regression fits, in that the estimated coefficients become exactly zero. In the Bayesian case the Bayes estimates do not provide exact annihilation of coefficients, but produce approximate sparseness (Park and Casella 2008).

At this point we note that, during the years following the appearance of Park and Casella (2008), several other proposals for sparse shrinkage of the β_j appeared in the literature (e.g. Armagan *et al.* 2013; Carvalho *et al.* 2010; Griffin and Brown 2011). Their accommodation in Infer.NET could also be entertained. Here we restrict attention to Laplace-based shrinkage.

We commence with the goal of fitting the following model in Infer.NET:

$$\begin{gathered}\boldsymbol{y}|\beta_0,\boldsymbol{\beta},\tau_\varepsilon \sim N(\boldsymbol{1}\beta_0+\boldsymbol{X\beta},\tau_\varepsilon^{-1}\boldsymbol{I}),\quad \beta_j|\,\tau_\beta \overset{\text{ind.}}{\sim} \text{Laplace}(0,\tau_\beta^{-1})\\ \beta_0\sim N(0,\tau_{\beta_0}^{-1}),\quad \tau_\beta\sim\text{Half-Cauchy}(A_\beta),\quad \tau_\varepsilon\sim\text{Half-Cauchy}(A_\varepsilon).\end{gathered} \tag{42}$$

The current release of Infer.NET does not support the Laplace distribution, so we require an auxiliary variable representation in a similar vein to what was used for the t-distribution in Section 5. We first note the distributional statement

$$x|\tau \sim \text{Laplace}(0,\tau^{-1}) \quad \text{if and only if} \quad x|g \sim N(0,g), \quad g|\tau \sim \text{Gamma}(1,\tfrac{1}{2}\,\tau), \tag{43}$$

which is exploited by Park and Casella (2008). However, auxiliary variable representation (43) is also not supported by Infer.NET, due to violation of its conjugacy rules. We get around this by working with the alternative auxiliary variable set-up:

$$x|\,c \sim N(0,1/c), \quad c|\,d \sim \text{Gamma}(M, M\,d), \quad d|\tau \sim \text{Gamma}(1,\tfrac{1}{2}\,\tau) \tag{44}$$

which is supported by Infer.NET, and leads to good approximation to the Laplace$(0,\tau^{-1})$ distribution when M is large. For any given $M > 0$, the density function of x satisfying auxiliary variable representation (44) is

$$\begin{aligned} p(x|\tau;M) &= \int_0^\infty \int_0^\infty (2\pi/c)^{-1/2}\exp\left(-\tfrac{1}{2}\,c\,x\right)\frac{(M\,d)^M}{\Gamma(M)}\,c^{M-1}\,\exp(-Mdc) \\ &\qquad \times \tfrac{1}{2}\tau\,\exp\left(-\tfrac{1}{2}\,d\tau\right)\,dc\,dd \\ &= \frac{\tau^{1/2}\,\Gamma\left(M+\frac{1}{2}\right)}{2\Gamma(M)\,M^{1/2}}\;{}_1F_1\left(M+\tfrac{1}{2};\tfrac{1}{2};\frac{\tau x^2}{4M}\right) - \tfrac{1}{2}\,\tau^{1/2}\,|x|\,{}_1F_1\left(M+1;3/2;\frac{\tau x^2}{4M}\right) \end{aligned}$$

where ${}_1F_1$ denotes the degenerate hypergeometric function (e.g. Gradshteyn and Ryzhik 1994). Figure 9 shows $p(x|\tau;M)$ for $\tau = 1$ and $M = 1, 10, 30$. The approximation to the Laplace$(0,1)$ density function is excellent for M as low as 10.

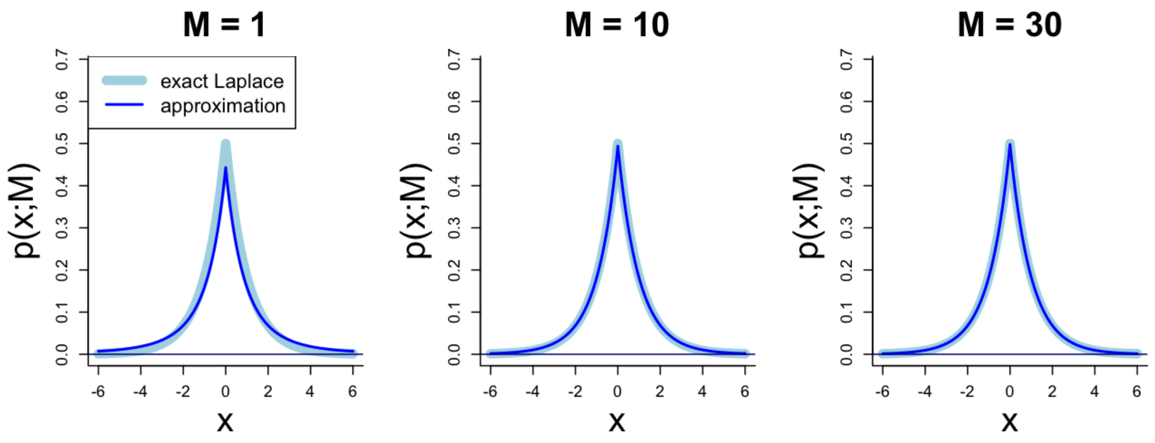


Figure 9: The exact Laplace$(0,1)$ density function $p(x) = \frac{1}{2}\exp(-|x|)$ and three approximations to it based on $p(x|\tau;M)$ for $\tau = 1$ and $M = 1, 10, 30$.

The convergence in distribution of random variables having the density function $p(x|\tau;M)$ to a Laplace$(0,\tau^{-1})$ random variable as $M \to \infty$ may be established using (27) and the fact that the family of t distributions tends to a Normal distribution as the degrees of freedom parameter increases, and then mixing this limiting distribution with a $\text{Gamma}(1, \frac{1}{2}\,\tau)$ variable.

In lieu of (42), the actual model fitted in Infer.NET is:

$$
\begin{gathered}
\boldsymbol{y}|\beta_0, \boldsymbol{\beta}, \tau_\varepsilon \sim N(\mathbf{1}\beta_0 + \boldsymbol{X}\boldsymbol{\beta}, \tau_\varepsilon^{-1}\boldsymbol{I}),\\
\boldsymbol{a}|\,\beta_0, \boldsymbol{\beta}, c_1, \ldots, c_p \sim N\left(\left[\begin{array}{c}\beta_0\\ \boldsymbol{\beta}\end{array}\right], \left[\begin{array}{cc}\tau_{\beta_0}^{-1} & \mathbf{0}\\ \mathbf{0} & \underset{1\le j\le p}{\text{diag}}\,(1/c_j)\boldsymbol{I}\end{array}\right]\right),\\
\left[\begin{array}{c}\beta_0\\ \boldsymbol{\beta}\end{array}\right] \sim N(\mathbf{0}, \kappa^{-1}\boldsymbol{I}), \quad c_j|\,d_j \overset{\text{ind.}}{\sim} \text{Gamma}(M, M\,d_j),\\
d_j|\tau_\beta \overset{\text{ind.}}{\sim} \text{Gamma}(1, \tfrac{1}{2}\tau_\beta), \quad \tau_\beta|b_\beta \sim \text{Gamma}(\tfrac{1}{2}, b_\beta), \quad b_\beta \sim \text{Gamma}(\tfrac{1}{2}, 1/A_\beta^2),\\
\tau_\varepsilon|\,b_\varepsilon \sim \text{Gamma}(\tfrac{1}{2}, b_\varepsilon), \quad b_\varepsilon \sim \text{Gamma}(\tfrac{1}{2}, 1/A_\varepsilon^2).
\end{gathered}
\tag{45}
$$

We use $\kappa = 10^{-10}, M = 100, \tau_{\beta_0} = 10^{-10}$, $A_\beta = A_\varepsilon = 10^5$.

Our illustration of (45) involves data from a diabetes study that was also used in Park and Casella (2008). The sample size is $n = 442$ and the number of predictor variables is $p = 10$. The response variable is a continuous index of disease progression one year after baseline. A description of the predictor variables is given in Efron *et al.* (2004). Their abbreviations, used in Park and Casella (2008), are: (1) age, (2) sex, (3), bmi, (4), map, (5), tc, (6) ldl, (7), hdl, (8), tch, (9) ltg, and (10) glu.

Figure 10 shows the fits obtained from both Infer.NET and MCMC. We see that the correspondence is very good, especially for the regression coefficients.

9. Measurement error model

In some situations some variables may be measured with error, i.e., the observed data are not the true values of those variables themselves, but a contaminated version of these variables. Examples of such situations include AIDS studies where CD4 counts are known to be measured with errors (Wu 2002) or air pollution data (Bachrach *et al.* 1999). Carroll *et al.* (2006) offers a comprehensive treatment of measurement error models. As described there, failure to account for measurement error can result in biased and misleading conclusions.

Let (x_i, y_i), $1 \le i \le n$, be a set of predictor/response pairs that are modeled according to

$$y_i|x_i, \beta_0, \beta_1, \tau_\varepsilon \overset{\text{ind.}}{\sim} N(\beta_0 + \beta_1 x_i, \tau_\varepsilon^{-1}), \quad 1 \le i \le n. \tag{46}$$

However, instead of observing the x_is we observe

$$w_i|x_i \overset{\text{ind.}}{\sim} N(x_i, \tau_z^{-1}), \quad 1 \le i \le n. \tag{47}$$

In other words, the predictors are measured with error with τ_z controlling the extent of the contamination. In general τ_z can be estimated through validation data, where some of the x_i values are observed, or replication data, where replicates of the w_i values are available. To

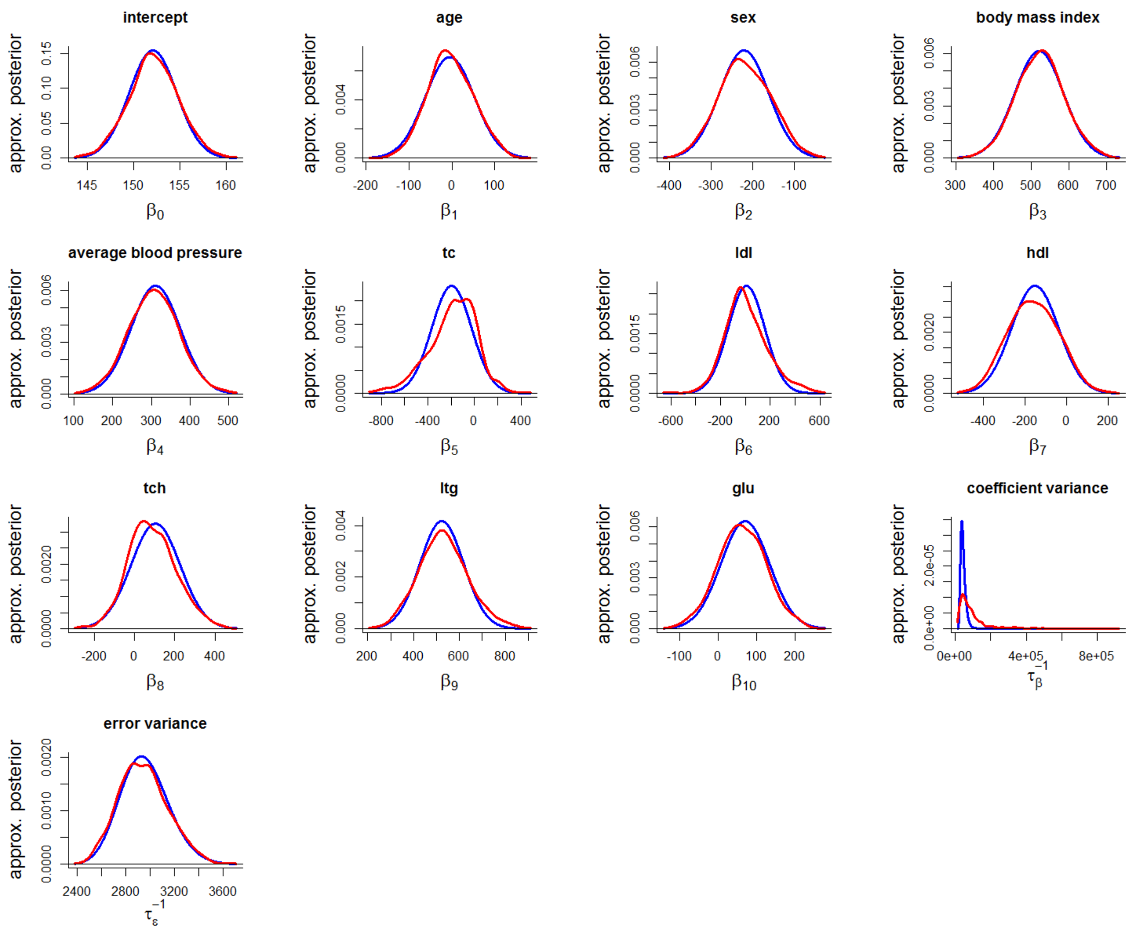


Figure 10: Approximate posterior density functions of model parameters in fitting (45) to data from a diabetes study.

simplify exposition we will assume that τ_z is known. Furthermore, we will assume that the x_is are Normally distributed:

$$x_i|\mu_x,\tau_x \overset{\text{ind.}}{\sim} N(\mu_x,\tau_x^{-1}), \quad 1 \le i \le n, \tag{48}$$

where μ_x and τ_x are additional parameters to be estimated.

Combining (46), (47) and (48) and, including aforementioned priors for variances, a Bayesian measurement error model can be represented by

$$\begin{gathered}\boldsymbol{y}|\,\boldsymbol{x},\boldsymbol{\beta},\tau_\varepsilon \sim N(\boldsymbol{X\beta},\tau_\varepsilon^{-1}\boldsymbol{I}), \quad \boldsymbol{w}|\boldsymbol{x} \sim N(\boldsymbol{x},\tau_z^{-1}\boldsymbol{I}), \quad \boldsymbol{x}|\mu_x,\tau_x \sim N(\mu_x\mathbf{1},\tau_x^{-1}\boldsymbol{I}),\\ \boldsymbol{\beta} \sim N(\mathbf{0},\tau_\beta^{-1}\boldsymbol{I}), \quad \tau_\varepsilon^{-1/2} \sim \text{Half-Cauchy}(A_\varepsilon), \\ \mu_x \sim N(0,\tau_{\mu_x}^{-1}) \quad \text{and} \quad \tau_x^{-1/2} \sim \text{Half-Cauchy}(A_x)\end{gathered} \tag{49}$$

where $\boldsymbol{\beta} = [\beta_0,\beta_1]$, $\boldsymbol{X} = [\mathbf{1},\boldsymbol{x}]$, τ_β, τ_μ, A_ε and A_x are user-specified constants and the vectors $\boldsymbol{y}$ and $\boldsymbol{w}$ are observed. We set $\tau_\beta = \tau_\mu = 10^{-10}$ and $A_\varepsilon = A_x = 10^5$.

Invoking (1), we arrive at the model

$$\begin{gathered}\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\beta},\tau_\varepsilon \sim N(\boldsymbol{X\beta},\tau_\varepsilon^{-1}\boldsymbol{I}), \quad \boldsymbol{w}|\boldsymbol{x}\sim N(\boldsymbol{x},\tau_z^{-1}\boldsymbol{I}), \quad \boldsymbol{x}|\,\mu_x,\tau_x\sim N(\mu_x\boldsymbol{1},\tau_x^{-1}\boldsymbol{I})\\ \boldsymbol{\beta}\sim N(\boldsymbol{0},\tau_\beta^{-1}\boldsymbol{I}), \quad \tau_\varepsilon|\,b_\varepsilon\sim \text{Gamma}(\tfrac{1}{2},1/b_\varepsilon), \quad b_\varepsilon\sim\text{Gamma}(\tfrac{1}{2},1/A_\varepsilon^2),\\ \mu_x\sim N(0,\tau_{\mu_x}^{-1}), \quad \tau_x|\,b_x\sim\text{Gamma}(\tfrac{1}{2},1/b_x) \quad\text{and}\quad b_x\sim\text{Gamma}(\tfrac{1}{2},1/A_x^2).\end{gathered} \tag{50}$$

Infer.NET 2.5, Beta 2 does not appear to be able to fit the model (50) under the product restriction

$$q(\boldsymbol{\beta},\tau_\varepsilon,b_\varepsilon,\boldsymbol{x},\mu_x,\tau_x,b_x)=q(\beta_0,\beta_1)\,q(\mu_x)\,q(\boldsymbol{x})\,q(\tau_\varepsilon,\tau_x)\,q(b_\varepsilon,b_x).$$

However, Infer.NET was able to fit this model with

$$q(\boldsymbol{\beta},\tau_\varepsilon,b_\varepsilon,\boldsymbol{x},\mu_x,\tau_x,b_x)=q(\beta_0)q(\beta_1)\,q(\mu_x)q(\boldsymbol{x})\,q(\tau_\varepsilon,\tau_x)\,q(b_\varepsilon,b_x). \tag{51}$$

Unfortunately, fitting under restriction (51) leads to poor accuracy. One possible remedy involves the centering transformation:

$$\widetilde{w}_i = w_i-\overline{w}.$$

Under this transformation, and with diffuse priors, the marginal posterior distributions of β_0 and β_1 are nearly independent.

Let $\widetilde{q}(\beta_0)$, $\widetilde{q}(\beta_1)$, $\widetilde{q}(\mu_x)$, $\widetilde{q}(\boldsymbol{x})$, $\widetilde{q}(\tau_\varepsilon,\tau_x)$ and $\widetilde{q}(b_\varepsilon,b_x)$ be the optimal values of $q(\beta_0)$, $q(\beta_1)$, $q(\mu_x)$, $q(\boldsymbol{x})$, $q(\tau_\varepsilon,\tau_x)$ and $q(b_\varepsilon,b_x)$ when the transformed $\widetilde{w}_i$s are used in place of the w_is. This corresponds to the transformation on the $\mathbf{X}$ matrix

$$\widetilde{\mathbf{X}}=\mathbf{XR}\qquad\text{where}\qquad \mathbf{R}=\begin{bmatrix}1&0\\-\overline{w}&1\end{bmatrix}.$$

Suppose that

$$\begin{gathered}\widetilde{q}(\beta_0)\sim N(\widetilde{\mu}_{q(\beta_0)},\widetilde{\sigma}^2_{q(\beta_0)}), \qquad \widetilde{q}(\beta_1)\sim N(\widetilde{\mu}_{q(\beta_1)},\widetilde{\sigma}^2_{q(\beta_1)}),\\ \widetilde{q}(\mu_x)\sim N(\widetilde{\mu}_{q(\mu_x)},\widetilde{\sigma}^2_{q(\mu_x)}), \quad\text{and}\quad \widetilde{q}(x_i)\sim N(\widetilde{\mu}_{q(x_i)},\widetilde{\sigma}^2_{q(x_i)}), \quad 1\le i\le n.\end{gathered}$$

Then we can back-transform approximately using

$$\begin{gathered}q(\beta_0,\beta_1)\sim N(\boldsymbol{\mu}_{q(\boldsymbol{\beta})},\boldsymbol{\Sigma}_{q(\boldsymbol{\beta})}), \qquad q(\mu_x)\sim N(\mu_{q(\mu_x)},\sigma^2_{q(\mu_x)}),\\ q(x_i)\sim N(\mu_{q(x_i)},\sigma^2_{q(x_i)}),\ 1\le i\le n,\end{gathered}$$

where

$$\begin{gathered}\boldsymbol{\mu}_{q(\boldsymbol{\beta})}=\mathbf{R}\begin{bmatrix}\widetilde{\mu}_{q(\beta_0)}\\ \widetilde{\mu}_{q(\beta_1)}\end{bmatrix}, \qquad \boldsymbol{\Sigma}_{q(\boldsymbol{\beta})}=\mathbf{R}\begin{bmatrix}\widetilde{\sigma}^2_{q(\beta_0)}&0\\0&\widetilde{\sigma}^2_{q(\beta_1)}\end{bmatrix}\mathbf{R}^T,\\ \mu_{q(\mu_x)}=\widetilde{\mu}_{q(\mu_x)}+\overline{w}, \qquad \sigma^2_{q(\mu_x)}=\widetilde{\sigma}^2_{q(\mu_x)},\\ \mu_{q(x_i)}=\widetilde{\mu}_{q(x_i)}+\overline{w}, \qquad \sigma^2_{q(x_i)}=\widetilde{\sigma}^2_{q(x_i)}, \qquad 1\le i\le n,\end{gathered}$$

$q(\tau_\varepsilon,\tau_x)=\widetilde{q}(\tau_\varepsilon,\tau_x)$ and $q(b_\varepsilon,b_x)=\widetilde{q}(b_\varepsilon,b_x)$.

To illustrate the use of Infer.NET we simulated data according to

$$x_i \overset{\text{ind.}}{\sim} N(1/2, 1/36), \quad w_i|\, x_i \overset{\text{ind.}}{\sim} N(x_i, 1/100) \quad \text{and} \quad y_i|x_i \overset{\text{ind.}}{\sim} N(0.3 + 0.7x_i, 1/16) \tag{52}$$

for $1 \leq i \leq n$ with $n = 50$. Results from a single simulation are illustrated in Figure 11. Similar results were obtained from other simulated data sets. From Figure 11 we see reasonable agreement of posterior density estimates produced by Infer.NET and MCMC for all parameters. Extension to the case where the mean of the y_is are modeled nonparametrically is covered in Pham *et al.* (2013). However, the current version of Infer.NET does not support models of this type. Such versatility limitations of Infer.NET are discussed in Section 11.1.

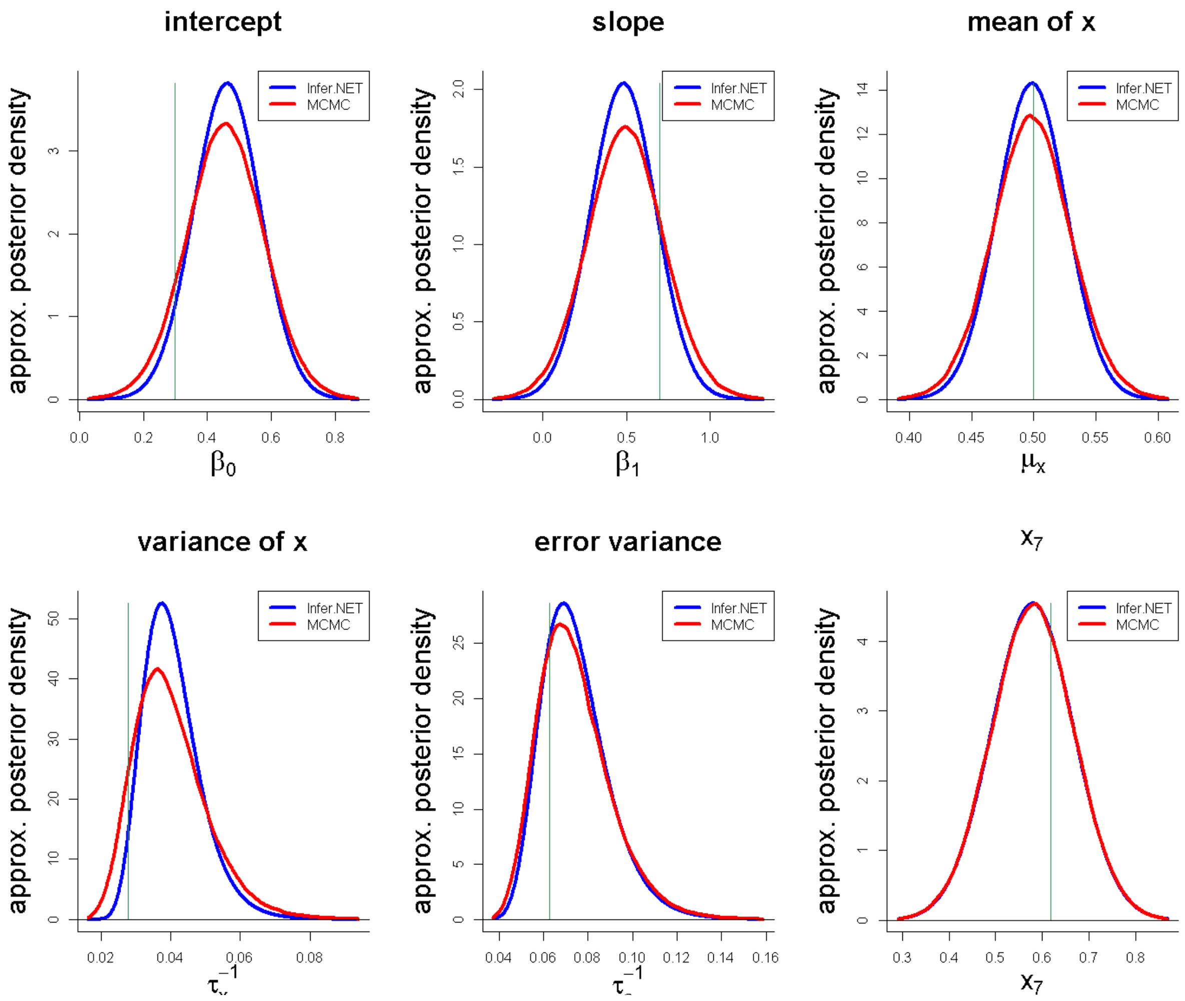


Figure 11: Variational Bayes approximate posterior density functions produced by Infer.NET for simulated data (52).

10. Missing predictor model

Missing data is a commonly occurring issue in statistical problems. Naïve methods for dealing with this issue, such as ignoring samples containing missing values, can be shown to be demonstrably optimistic (i.e., standard errors are deflated), biased or simply an inefficient use of the data. Little and Rubin (2002) contains a detailed discussion of the various issues at stake.

Consider a simple linear regression model with response y_i with predictor x_i:

$$y_i|x_i,\beta_0,\beta_1,\tau_\varepsilon \overset{\text{ind.}}{\sim} N(\beta_0+\beta_1 x_i,\tau_\varepsilon^{-1}),\quad 1\le i\le n. \tag{53}$$

Suppose that some of the x_is are missing and let r_i be an indicator x_i being observed. Specifically,

$$r_i=\begin{cases}1 & \text{if } x_i \text{ is observed,}\\ 0 & \text{if } x_i \text{ is missing}\end{cases}\qquad \text{for } 1\le i\le n.$$

A missing data model contains at least two components:

- A model for the underlying distribution of the variable that is subject to missingness data. We will use

$$x_i|\mu_x,\tau_x \overset{\text{ind.}}{\sim} N(\mu_x,\tau_x^{-1}),\quad 1\le i\le n. \tag{54}$$

- A model for the missing data mechanism, which models the probability distribution of r_i.

As detailed in Faes *et al.* (2011), there are several possible models, with varying degrees of sophistication, for the missing data mechanism. Here the simplest such model, known as *missing completely at random*, is used:

$$\begin{gathered}\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\beta},\tau_\varepsilon\sim N(\boldsymbol{X\beta},\tau_\varepsilon^{-1}\boldsymbol{I}),\quad \boldsymbol{x}|\,\mu_x,\tau_x\sim N(\mu_x\boldsymbol{1},\tau_x^{-1}\boldsymbol{I}),\quad r_i\overset{\text{ind.}}{\sim}\text{Bernoulli}(p),\\ \boldsymbol{\beta}\sim N(\boldsymbol{0},\tau_\beta^{-1}\boldsymbol{I}),\qquad \tau_\varepsilon|\,b_\varepsilon\sim\text{Gamma}(\tfrac{1}{2},1/b_\varepsilon),\qquad b_\varepsilon\sim\text{Gamma}(\tfrac{1}{2},1/A_\varepsilon^2),\\ \mu_x\sim N(0,\tau_\mu^{-1}),\qquad \tau_x\,|\,b_x\sim\text{Gamma}(\tfrac{1}{2},b_x)\qquad\text{and}\qquad b_x\sim\text{Gamma}(\tfrac{1}{2},1/A_x^2),\end{gathered} \tag{55}$$

where $\boldsymbol{\beta}=[\beta_0,\beta_1]$, $\boldsymbol{X}=[\boldsymbol{1},\boldsymbol{x}]$, τ_β, τ_μ, A_ε and A_x are user-specified hyperparameters. Again, we will use $\tau_\beta=\tau_\mu=10^{-10}$ and $A_\varepsilon=A_x=10^5$.

The simulated data that we use in our illustration of Infer.NET involves the following sample size and 'true values':

$$n=50,\quad \beta_0=0.3,\quad \beta_1=0.7,\quad \mu_x=0.5,\quad \tau_x=36,\quad\text{and}\quad p=0.75. \tag{56}$$

Putting $p=0.75$ implies that, on average, 25% of the predictor values are missing completely at random.

Infer.NET 2.5, Beta 2 is not able to fit the model (50) under the product restriction

$$q(\boldsymbol{\beta},\tau_\varepsilon,b_\varepsilon,\boldsymbol{x}_{\text{mis}},\mu_x,\tau_x,b_x)=q(\beta_0,\beta_1)\,q(\boldsymbol{x}_{\text{mis}})\,q(\tau_\varepsilon,\tau_x)\,q(b_\varepsilon,b_x), \tag{57}$$

where $\boldsymbol{x}_{\text{mis}}$ denotes the vector of missing predictors, but is able to handle

$$q(\boldsymbol{\beta},\tau_\varepsilon,b_\varepsilon,\boldsymbol{x}_{\text{mis}},\mu_x,\tau_x,b_x)=q(\beta_0)\,q(\beta_1)\,q(\boldsymbol{x}_{\text{mis}})\,q(\tau_\varepsilon,\tau_x)\,q(b_\varepsilon,b_x), \tag{58}$$

and was used in our illustration of Infer.NET for such models.

We also used BUGS to perform Bayesian inference via MCMC as a benchmark for comparison of results. For similar reasons as in the previous section fitting under this restriction (58) leads to poor results. Instead we perform the centering transformation

$$\widetilde{x}_i = x_i - \overline{x}_{\text{obs}}$$

where $\overline{x}_{\text{obs}}$ is the mean of the observed x values. This transformation makes the marginal posterior distributions of β_0 and β_1 nearly independent.

Let $\widetilde{q}(\beta_0)$, $\widetilde{q}(\beta_1)$, $\widetilde{q}(\mu_x)$, $\widetilde{q}(\boldsymbol{x}_{\text{mis}})$, $\widetilde{q}(\tau_\varepsilon,\tau_x)$ and $\widetilde{q}(b_\varepsilon,b_x)$ be the optimal values of $q(\beta_0)$, $q(\beta_1)$, $q(\mu_x)$, $q(\boldsymbol{x}_{\text{mis}})$, $q(\tau_\varepsilon,\tau_x)$ and $q(b_\varepsilon,b_x)$ when the transformed $\widetilde{x}_i$s are used in place of the x_is. This corresponds to the transformation on the $\mathbf{X}$ matrix

$$\widetilde{\mathbf{X}} = \mathbf{X}\mathbf{R} \qquad \text{where} \qquad \mathbf{R}=\begin{bmatrix} 1 & 0 \\ -\overline{x}_{\text{obs}} & 1 \end{bmatrix}.$$

Suppose that

$$\widetilde{q}(\beta_0)\sim N(\widetilde{\mu}_{q(\beta_0)},\widetilde{\sigma}^2_{q(\beta_0)}), \qquad \widetilde{q}(\beta_1)\sim N(\widetilde{\mu}_{q(\beta_1)},\widetilde{\sigma}^2_{q(\beta_1)}),$$
$$\widetilde{q}(\mu_x)\sim N(\widetilde{\mu}_{q(\mu_x)},\widetilde{\sigma}^2_{q(\mu_x)}), \quad \text{and} \qquad \widetilde{q}(x_i)\sim N(\widetilde{\mu}_{q(x_{\text{mis},i})},\widetilde{\sigma}^2_{q(x_{\text{mis},i})}),\ 1\le i\le n_{\text{mis}},$$

then we back-transform approximately using

$$q(\beta_0,\beta_1)\sim N(\boldsymbol{\mu}_{q(\boldsymbol{\beta})},\boldsymbol{\Sigma}_{q(\boldsymbol{\beta})}), \qquad q(\mu_x)\sim N(\mu_{q(\mu_x)},\sigma^2_{q(\mu_x)}),$$
$$q(x_{\text{mis},i})\sim N(\mu_{q(x_{\text{mis},i})},\sigma^2_{q(x_{\text{mis},i})}),\ 1\le i\le n,$$

where

$$\boldsymbol{\mu}_{q(\boldsymbol{\beta})}=\mathbf{R}\begin{bmatrix}\widetilde{\mu}_{q(\beta_0)}\\ \widetilde{\mu}_{q(\beta_1)}\end{bmatrix}, \qquad \boldsymbol{\Sigma}_{q(\boldsymbol{\beta})}=\mathbf{R}\begin{bmatrix}\widetilde{\sigma}^2_{q(\beta_0)} & 0\\ 0 & \widetilde{\sigma}^2_{q(\beta_1)}\end{bmatrix}\mathbf{R}^T,$$
$$\mu_{q(\mu_x)}=\widetilde{\mu}_{q(\mu_x)}+\overline{x}_{\text{obs}}, \qquad \sigma^2_{q(\mu_x)}=\widetilde{\sigma}^2_{q(\mu_x)},$$
$$\mu_{q(x_{\text{mis},i})}=\widetilde{\mu}_{q(x_{\text{mis},i})}+\overline{x}_{\text{obs}}, \qquad \sigma^2_{q(x_{\text{mis},i})}=\widetilde{\sigma}^2_{q(x_{\text{mis},i})}, \qquad 1\le i\le n,$$

$q(\tau_\varepsilon,\tau_x)=\widetilde{q}(\tau_\varepsilon,\tau_x)$ and $q(b_\varepsilon,b_x)=\widetilde{q}(b_\varepsilon,b_x)$.

Results from a single simulation are illustrated in Figure 12. Similar results were obtained from other simulated data sets. From Figure 12 we see reasonable agreement of posterior density estimates produced by Infer.NET and MCMC for all parameters. Extension to the case where the mean of y is modeled nonparametrically is covered in Faes *et al.* (2011). However, the current version of Infer.NET does not support models of this type.

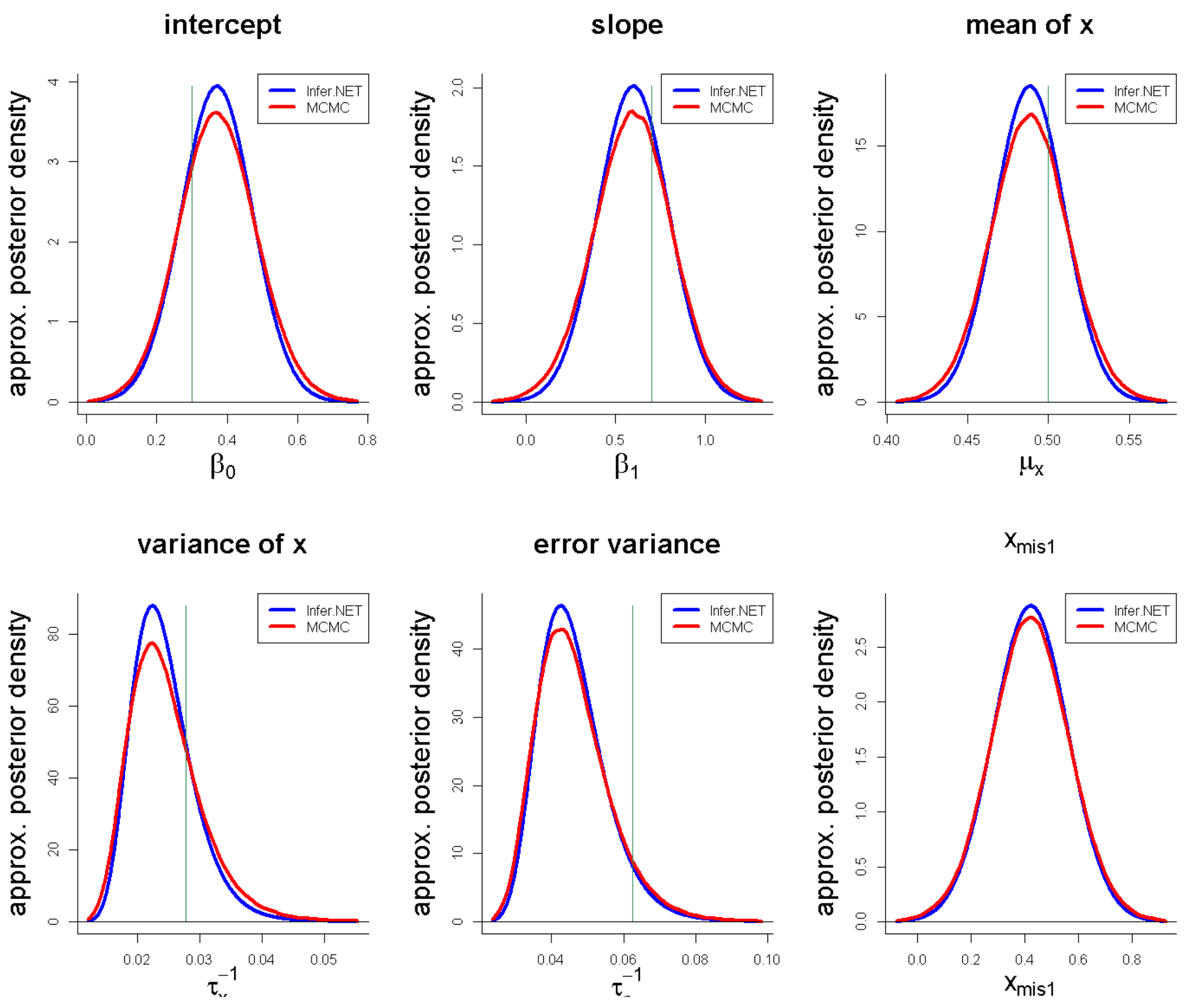


Figure 12: Variational Bayes approximate posterior density functions produced by Infer.NET data simulated according to (56).

11. Comparison with BUGS

We now make some comparisons between Infer.NET and BUGS in the context of Bayesian semiparametric regression.

11.1. Versatility comparison

These comments mainly echo those given in Section 5 of Wang and Wand (2011), so are quite brief. BUGS is much more versatile than Infer.NET, with the latter subject to restrictions such as standard distributional forms, conjugacy rules and not being able to handle models such as (13) without the trick manifest in (15). The measurement error and missing data regression examples given in Sections 9 and 10 are feasible in Infer.NET for *parametric* regression, but not for *semiparametric* regression such as the examples in Sections 4, 5 and 8 of Marley and Wand (2010). Nevertheless, as demonstrated in this article, there is a wide variety of semiparametric regression models that can be handled by Infer.NET.

11.2. Accuracy comparison

In general, BUGS can always be more accurate than Infer.NET since MCMC suffers only from Monte Carlo error, which can be made arbitrarily small via larger sample sizes. The Infer.NET inference engines, expectation propagation and variational message passing, have inherent approximation errors that cannot be completely eliminated. However, our examples show that the accuracy of Infer.NET is quite good for a wide variety of semiparametric regression models.

11.3. Timing comparison

Table 2 gives some indication of the relative computing times for the examples in the paper. In this table we report the average elapsed (and standard error) of the computing times over 100 runs with the number of variational Bayes or expectation propagation iterations set to 100 and the MCMC sample sizes set at 10000. This was sufficient for convergence in these particular examples. All examples were run on the third author's laptop computer (64 bit Windows 8 Intel i7-4930MX central processing unit at 3GHz with 32GB of random access memory). We concede that comparison of deterministic and Monte Carlo algorithms is fraught with difficulties. However, convergence criteria aside, these times give an indication of the performance in terms of the implementations of each of the algorithms for particular models on a fast 2014 computer. Note that we did not fit the geoadditive model in Section 7 via BUGS due to the excessive time required.

section number	semiparametric regression model	time in seconds for Infer.NET	time in seconds for BUGS
3	Simple semiparametric regression	1.55 (0.01)	8.60 (0.01)
4	Generalized additive model (logistic)	3.09 (0.01)	81.30 (0.13)
4	Generalized additive model (probit)	25.80 (0.07)	79.85 (0.04)
5	Robust nonparametric regression	80.38 (0.03)	22.56 (0.06)
6	Semiparametric mixed model	666.73 (0.56)	477.59 (0.06)
7	Geoadditive model	1231.12 (3.05)	—— ——
8	Bayesian lasso regression	1.80 (0.01)	199.59 (0.30)
9	Measurement error model	1.50 (0.01)	7.05 (0.02)
10	Missing predictor model	1.53 (0.01)	5.51 (0.03)

Table 2: Average run times (standard errors) in seconds over 100 runs of the methods for each of the examples in the paper.

Table 2 reveals that Infer.NET offers considerable speed-ups compared with BUGS for most of the examples. The exceptions are the robust nonparametric regression model of Section 5 and the semiparametric mixed model of Section 6. The slowness of the robust nonparametric regression fit is mainly explained by the multiple calls to Infer.NET, corresponding to the degrees of freedom grid. The semiparametric mixed model is quite slow in the current release of Infer.NET due to the full sparse design matrices being carried around in the calculations. Recently developed streamlined approaches to variational inference for longitudinal and multilevel data analysis (Lee and Wand 2014) offer significant speed-ups. The geoadditive model of Section 7 is also slow to fit, with the large design matrices being a likely reason.

12. Summary

Through several examples, the efficacy of Infer.NET for semiparametric regression analysis has been demonstrated. Generally speaking, the fitting and inference is shown to be quite fast and accurate. Models with very large design matrices are an exception and can take significant amounts of time to fit using the current release of Infer.NET.

Our survey of Infer.NET in the context of semiparametric regression will aid future analyses of this type via fast graphical models software, by building upon the examples presented here. It may also influence future directions for the development of fast graphical models methodology and software.

Appendix: Details of Infer.NET fitting of a simple semiparametric model

This section provides an extensive description of the Infer.NET program for semiparametric regression analysis of the Onions data set based on model (15) in Section 3. The data are first transformed as explained in Section 3 and these transformed versions are represented by $\boldsymbol{y}$ and $\boldsymbol{C} = [\boldsymbol{X}\ \boldsymbol{Z}]$. Thereafter, the text files `K.txt`, `y.txt`, `Cmat.txt`, `sigsqBeta.txt`, `Au.txt`, `Aeps.txt` and `nIterVB.txt` are generated. The first three files contain the number of spline basis functions, $\boldsymbol{y}$ and $\boldsymbol{C}$, respectively. The following three text files each contain a single number and represent the values for the hyperparameters: $\tau_\beta = 10^{-10}$, $A_\varepsilon = 10^5$ and $A_u = 10^5$. The last file, `nIterVB.txt`, contains a single positive number (i.e., 100) that specifies the number of mean field variational Bayes iterations. All these files were set up in R.

The actual Infer.NET code is a C♯ script, which can be run from Visual Studio 2010 or DOS within the Microsoft Windows operating system. The following paragraphs explain the different commands in the C♯ script. Firstly, all required C♯ and Infer.NET libraries are loaded:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using System.Reflection;
using MicrosoftResearch.Infer;
using MicrosoftResearch.Infer.Distributions;
using MicrosoftResearch.Infer.Maths;
using MicrosoftResearch.Infer.Models;
```

Next, the full path name of the directory containing the above-mentioned input files needs to be specified:

```
string dir = "C:\\research\\inferNet\\onions\\";
```

The values for the hyperparameters, number of variational Bayes iterations and number of spline basis functions are imported using the commands:

```
double tauBeta = new DataTable(dir+"tauBeta.txt").DoubleNumeric;
double Au = new DataTable(dir+"Au.txt").DoubleNumeric;
double Aeps = new DataTable(dir+"Aeps.txt").DoubleNumeric;
int nIterVB = new DataTable(dir+"nIterVB.txt").IntNumeric;
int K = new DataTable(dir+"K.txt").IntNumeric;
```

Reading in the data $\boldsymbol{y}$ and $\boldsymbol{C}$ proceeds as follows:

```
DataTable yTable = new DataTable(dir+"y.txt");
DataTable C = new DataTable(dir+"Cmat.txt");
```

These commands make use of the C♯ object class `DataTable`. This class, which was written by the authors, facilitates the input of general rectangular arrays of numbers when in a text file. The following line extracts the number of observations from the data matrix $\boldsymbol{C}$:

```
int n = C.numRow;
```

The observed values of $\boldsymbol{C}$ and $\boldsymbol{y}$, which are stored in objects `C` and `yTable`, are assigned to the objects `cvec` and `y` via the next set of commands:

```
Vector[] cvecTable = new Vector[n];
for (int i = 0; i < n; i++)
{
   cvecTable[i] = Vector.Zero(3+K);
   for (int j = 0; j < 3+K; j++)
      cvecTable[i][j] = C.DataMatrix[i,j];
}

Range index = new Range(n).Named("index");
VariableArray<Vector> cvec = Variable.Array<Vector>(
   index).Named("cvec");
cvec.ObservedValue = cvecTable;
VariableArray<double> y = Variable.Array<double>(index).Named("y");
y.ObservedValue = yTable.arrayForIN;
```

The function `arrayForIN` is member of the class `DataTable` and stores the data as an array to match the type of `y.ObservedValue`. The resulting objects are now directly used to specify the prior distributions. First, the commands in code chunk (16) in Section 3 are used to set up priors for τ_u and τ_ε, while the trick involving the auxiliary variable $\boldsymbol{a} \equiv \boldsymbol{0}$ in (15) is coded as:

```
Vector zeroVec = Vector.Zero(3+K);
PositiveDefiniteMatrix betauPrecDummy =
   new PositiveDefiniteMatrix(3+K,3+K);
for (int j = 0; j < 3+K; j++)
   betauPrecDummy[j,j] = kappa;
Variable<Vector> betauWork =
   Variable.VectorGaussianFromMeanAndPrecision(zeroVec,
   betauPrecDummy).Named("betauWork");

Variable<double>[] betauDummy = new Variable<double>[3+K];
for (int j = 0; j < 3+K; j++)
   betauDummy[j] = Variable.GetItem(betauWork,j);

Variable<double>[] a = new Variable<double>[3+K];
for (int j = 0; j < 3; j++)
{
   a[j] = Variable.GaussianFromMeanAndVariance(betauDummy[j],
      tauBeta);
   a[j].ObservedValue = 0.0;
}
for (int k = 0 ; k < K; k++)
{
   a[3+k] = Variable.GaussianFromMeanAndPrecision(betauDummy[3+k],
      tauU);
   a[3+k].ObservedValue = 0.0;
}
```

The command to specify the likelihood for model (15) is listed as code chunk (17) in Section 3. The inference engine and the number of mean field variational Bayes iterations are specified by means of code chunk (18). Finally, the following commands write the estimated values for the parameters of the approximate posteriors to files named `mu.q.betau.txt`, `Sigma.q.betau.txt`, `parms.q.tauEps.txt` and `parms.q.tauu.txt`:

```
SaveData.SaveTable(engine.Infer<VectorGaussian>(betauWork).GetMean(),
   dir+"mu.q.betau.txt");
SaveData.SaveTable(engine.Infer<VectorGaussian>(
   betauWork).GetVariance(),dir+"Sigma.q.betau.txt");
SaveData.SaveTable(engine.Infer<Gamma>(tauEps),
   dir+"parms.q.tauEps.txt");
SaveData.SaveTable(engine.Infer<Gamma>(tauu),dir+"parms.q.tauu.txt");
```

These commands involve the C♯ function named `SaveTable`, which is a method in the class `SaveData`. We wrote `SaveTable` to facilitate writing the output from Infer.NET to a text file. Summary plots, such as Figure 1, can be made in R after back-transforming the approximate posterior density parameters.

Acknowledgments

This research was partially funded by the Australian Research Council Discovery Project DP110100061. We are grateful to Andrew Chernih for his provision of the Sydney property

data.

References

Armagan A, Dunson D, Lee J (2013). "Generalized Double Pareto Shrinkage." *Statistica Sinica*, **23**, 119–143.

Bachrach LK, Hastie T, Wang MC, Narasimhan B, Marcus R (1999). "Bone Mineral Acquisition in Healthy Asian, Hispanic, Black, and Caucasian Youth: A Longitudinal Study." *Journal of Clinical Endocrinology & Metabolism*, **84**(12), 4702–4712.

Carroll RJ, Ruppert D, Stefanski LA, Crainiceanu C (2006). *Measurement Error in Nonlinear Models: A Modern Perspective*. 2nd edition edition. Chapman and Hall, London.

Carvalho CM, Polson NG, Scott J (2010). "The Horseshoe Estimator for Sparse Signals." *Biometrika*, **97**, 465–480.

Efron B, Hastie T, Johnstone I, Tibshirani R (2004). "Least Angle Regression." *The Annals of Statistics*, **32**, 407–451.

Faes C, Ormerod JT, Wand MP (2011). "Variational Bayesian Inference for Parametric and Nonparametric Regression with Missing Data." *Journal of the American Statistical Association*, **106**, 959–971.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models." *Bayesian Analysis*, **1**, 515–533.

Gradshteyn I, Ryzhik I (1994). *Table of Integrals, Series, and Products*. Academic Press, Boston.

Griffin J, Brown P (2011). "Bayesian Hyper-Lassos with Non-Convex Penalization." *Australian and New Zealand Journal of Statistics*, **53**, 423–442.

Haberman S (1976). "Generalized Residuals for Log-Linear Models." In *Proceedings of the 9th International Biometrics Conference*, pp. 104–122. Boston, USA.

Kammann EE, Wand MP (2003). "Geoadditive Models." *Journal of the Royal Statistical Society C*, **52**, 1–18.

Kaufman L, Rousseeuw PJ (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York.

Knowles DA, Minka TP (2011). "Non-Conjugate Message Passing for Multinomial and Binary Regression." *Advances in Neural Information Processing Systems 24*, pp. 1701– 1709.

Lange KL, Little RJA, Taylor JMG (1989). "Robust Statistical Modeling Using the t-Distribution." *The Journal of the American Statistical Association*, **84**, 881–896.

Lee Y, Wand M (2014). "Streamlined Mean Field Variational Bayes for Longitudinal and Multilevel Data Analysis." *In progress*.

Little RJA, Rubin DB (2002). *Statistical Analysis with Missing Data*. 2nd edition edition. Wiley, New York.

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

Marley JK, Wand MP (2010). "Non-Standard Semiparametric Regression Via BRugs." *Journal of Statistical Software*, **37**, 1–30.

Minka T (2001). "Expectation Propagation for Approximate Bayesian Inference." *Proceedings of Conference on Uncertainty in Artificial Intelligence*, pp. 51–76.

Minka T (2005). "Divergence Measures And Message Passing." *Microsoft Research Technical Report Series*, **MSR-TR-2008-173**, 1–17.

Minka T, Winn J (2008). "Gates: A Graphical Notation for Mixture Models." *Microsoft Research Technical Report Series*, **MSR-TR-2008-185**, 1–16.

Minka T, Winn J, Guiver J, Knowles D (2013). ***Infer.NET* 2.5**. URL `http://research.microsoft.com/infernet`.

Ormerod JT, Wand MP (2010). "Explaining Variational Approximations." *The American Statistician*, **64**, 140–153.

Park T, Casella G (2008). "The Bayesian Lasso." *Journal of the American Statistical Association*, **103**, 681–686.

Pham T, Ormerod JT, Wand MP (2013). "Mean Field Variational Bayesian Inference for Nonparametric Regression with Measurement Error." *Computational Statistics and Data Analysis*, **68**, 375–387.

Ruppert D, Wand MP, Carroll RJ (2003). *Semiparametric Regression*. Cambridge University Press, New York.

Ruppert D, Wand MP, Carroll RJ (2009). "Semiparametric Regression During 2003–2007." *Bayesian Analysis*, **3**, 1193–1265.

Saul LK, Jordan MI (1996). "Exploiting Tractable Substructures in Intractable Networks." In *Advances in Neural Information Processing Systems*, pp. 435–442. MIT Press, Cambridge, Massachusetts.

Staudenmayer J, Lake EE, Wand MP (2009). "Robustness for General Design Mixed Models Using the t-Distribution." *Statistical Modelling*, **9**, 235–255.

Tibshirani R (1996). "Regression Shrinkage and Selection Via the Lasso." *Journal of the Royal Statistical Society, Series B*, **58**, 267–288.

Wainwright M, Jordan MI (2008). "Graphical Models, Exponential Families, and Variational Inference." *Foundations and Trends in Machine Learning*, **1**, 1–305.

Wand MP (2009). "Semiparametric Regression and Graphical Models." *Australian and New Zealand Journal of Statistics*, **51**, 9–41.

Wand MP, Ormerod JT (2008). "On O'Sullivan Penalised Splines and Semiparametric Regression." *Australian and New Zealand Journal of Statistics*, **50**, 179–198.

Wand MP, Ormerod JT, Padoan SA, Frühwirth R (2011). "Mean Field Variational Bayes for Elaborate Distributions." *Bayesian Analysis*, **6**(4), 847–900.

Wand MP, Ripley BD (2010). *KernSmooth 2.23. Functions for Kernel Smoothing Corresponding to the Book: Wand, M.P. and Jones, M.C. (1995) Kernel Smoothing*. URL `http://cran.r-project.org`.

Wang SSJ, Wand MP (2011). "Using Infer.NET for Statistical Analyses." *The American Statistician*, **65**, 115–126.

Winn J, Bishop CM (2005). "Variational Message Passing." *Journal of Machine Learning Research*, **6**, 661–694.

Wu L (2002). "A Joint Model for Nonlinear Mixed-Effects Models with Censored Response and Covariates Measured with Error, with Application to AIDS Studies." *Journal of the American Statistical Association*, **97**, 700–709.

Affiliation:

Jan Luts
TheSearchParty.com Pty Ltd.
Level 1, 79 Commonwealth Street,
Surry Hills 2011, Australia
E-mail: jan@thesearchparty.com

Shen S.J. Wang
Department of Mathematics
The University of Queensland
Brisbane 4072, Australia
E-mail: s.wang7@uq.edu.au

John T. Ormerod
School of Mathematics and Statistics
University of Sydney
Sydney 2006, Australia
E-mail: jormerod@sydney.edu.au
URL: http://www.maths.usyd.edu.au/u/jormerod/

Matt P. Wand
School of Mathematical Sciences
University of Technology, Sydney
Broadway 2007, Australia
E-mail: matt.wand@uts.edu.au
URL: http://matt-wand.utsacademics.info

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd