Bayesian Analysis for Penalized Spline Regression Using WinBUGS

Ciprian M. Crainiceanu* David Ruppert† M.P. Wand‡

March 18, 2004

Abstract

Penalized splines can be viewed as BLUPs in a mixed model framework, which allows the use of mixed model software for smoothing. Thus, software originally developed for Bayesian analysis of mixed models can be used for penalized spline regression. Bayesian inference for nonparametric models enjoys the flexibility of nonparametric models and the exact inference provided by the Bayesian inferential machinery. This paper provides a simple, yet comprehensive, set of programs for the implementation of nonparametric Bayesian analysis in WinBUGS.

Keywords: MCMC, Semiparametric regression, Software

1 Introduction

The virtues of nonparametric regression models have been discussed extensively in the statistics literature. Competing approaches to nonparametric modeling include, but are not limited to, smoothing splines (Eubank [9]; Wahba [27]; Green and Silverman [14]), series-based smoothers (Tarter and Lock [25]; Ogden [20]), kernel methods (Wand and Jones [29]; Fan and Gijbels [10]); regression splines (Hastie and Tibshirani [17]; Friedman [11]; Hansen and Kooperberg [16]; penalized splines (Eilers and Marx, [7], Ruppert, Wand and Carroll [22]). The main advantage of nonparametric over parametric models is their flexibility. In the nonparametric framework the shape of the

*Department of Biostatistics, Johns Hopkins University, 615 N. Wolfe St. E3037 Baltimore, MD 21205 USA. E-mail: ccrainic@jhsph.edu

†School of Operational Research and Industrial Engineering, Cornell University, Rhodes Hall, NY 14853, USA. E-mail: ruppert@orie.cornell.edu

‡Department of Statistics, School of Mathematics, University of New South Wales, Sydney 2052, Australia E-mail: wand@maths.unsw.edu.au

functional relationship between covariates and the dependent variables is determined by the data, whereas in the parametric framework the shape is determined by the model.

In this paper we focus on semiparametric regression models using penalized splines (Ruppert, Wand and Carroll [22]), but the methodology can be extended to other penalized likelihood models. It is becoming more widely appreciated that penalized likelihood models can be viewed as particular cases of Generalized Linear Mixed Models (GLMMs): Eilers and Marx [7]; Brumback, Ruppert and Wand 1999, [3]; Ruppert, Wand and Carroll [22]. We discuss this in more details in section 2. Given this equivalence, statistical software developed for mixed models, such as S-plus (function `lme`) or SAS (`PROC MIXED` and the `GLIMMIX` macro) can be used for smoothing (Wand [28], Ngo and Wand [19]). There are at least two potential problems when using such software for inference in mixed models. Firstly, in the case of GLMMs the likelihood of the model is a high dimensional integral over the unobserved random effects and, in general, cannot be computed exactly and has to be approximated. This approximating character can have a sizeable effect on parameter estimation, especially on the variance components. The second problem is that confidence intervals are obtained by replacing the estimated parameters instead of the true parameters and ignoring the inherent additional variability. This results in tighter than normal confidence intervals and could be avoided by using bootstrap. However, standard software does not have bootstrap capabilities and favors the "plug-in" method.

Mixed models from classical statistics can be viewed as an intermediate step between frequentist and Bayesian models since some parameters are treated as random. Bayesian analysis treats all parameters as random, assigns prior distributions to characterize knowledge about parameter values prior to data collection, and uses the joint posterior distribution of parameters given the data as the basis of inference. The posterior density for complex models, like the likelihood function, is analytically unavailable. However, recent advances in simulation techniques, such as Markov Chain Monte Carlo (MCMC) have made possible the simulation of the posterior distribution for practically any model whose probability density function is known up to a normalizing constant. Moreover, the posterior distribution of any explicit function of the model parameters

can be obtained as a by-product of the simulation algorithm.

The Bayesian inference for nonparametric models enjoys the flexibility of nonparametric models and the exact inference provided by the Bayesian inferential machinery. It is this combination that makes Bayesian nonparametric modeling so attractive (e.g. Berry, Carroll, and Ruppert [2]; Ruppert, Wand and Carroll [22]). These approaches provide the general methodology and expert software for Bayesian inference.

The goal of this paper is not to discuss Bayesian methodology, nonparametric regression or provide novel modeling techniques. Instead, we provide a simple, yet comprehensive, set of programs for the implementation of nonparametric Bayesian analysis in WinBUGS (Spiegelhalter, Thomas and Best [24]), which has become the standard software for Bayesian analysis. We hope to win over practitioners who were not previously using these methods due to technical difficulties. Following the methodology described by Ruppert, Wand and Carroll [22] we will not concern ourselves with Bayesian knot selection. Rather, for any given application, the knots are fixed, being usually chosen at the sample quantiles of the covariates, as described by Ruppert [21].

2 Penalized splines as mixed models

Ruppert, Wand and Carroll [22] present the general methodology of semiparametric modeling using the inferential equivalence between penalized likelihood models (which include penalized splines) with mixed models. The authors underline the modularity of their approach which allows covariates to enter the model parametrically or nonparametrically and provide S+, MATLAB and WinBUGS software for inference using semiparametric models. Ngo and Wand [19] provide S-plus and SAS software for the "frequentist" analysis of a comprehensive set of examples of semiparametric models. This paper shows how to do the Bayesian analysis of semiparametric models using WinBUGS.

In this section we explain why one can view penalized splines as mixed models. Consider the regression model

$$y_i = m\left(x_i\right) + \epsilon_i \ ,$$

where ϵ_i are i.i.d. $N\left(0, \sigma_\epsilon^2\right)$, ϵ_i is independent x_i, and $m(\cdot)$ is a smooth function. We

consider the x's to be random but their joint distribution is unspecified and could be degenerate, e.g., the x's could be equally spaced. The smooth function can be modeled, for example, using the class of spline functions

$$m\left(x,\boldsymbol{\theta}\right)=\beta_0+\beta_1 x+\ldots+\beta_p x^p+\sum_{k=1}^{K}b_k\left(x-\kappa_k\right)_+^p\ ,$$

where $\boldsymbol{\theta}=\left(\beta_1,\ldots,\beta_p,b_1,\ldots,b_K\right)^T$ is the vector of regression coefficients, and $\kappa_1<\kappa_2<\ldots<\kappa_K$ are fixed knots. Here a_+ is equal to a if a is positive and zero otherwise. Also, $a_+^p=(a_+)^p$. Following Ruppert (2002), we consider a number of knots that is large enough (typically 5 to 20) to ensure the desired flexibility, and κ_k is the sample quantile of x's corresponding to probability $k/(K+1)$, but results hold for any other choice of knots. To avoid overfitting, we minimize

$$\sum_{i=1}^{n}\left\{y_i-m\left(x_i,\boldsymbol{\theta}\right)\right\}^2+\frac{1}{\lambda}\boldsymbol{\theta}^T\boldsymbol{D}\boldsymbol{\theta}\ , \tag{1}$$

where λ is the smoothing parameter and $\boldsymbol{D}$ is a known positive semi-definite penalty matrix. The penalty $\int\left\{m^{(2)}(x,\boldsymbol{\theta})\right\}^2dx$ used for smoothing splines can be achieved with $\boldsymbol{D}$ equal to the sample second moment matrix of the second derivatives of the spline basis functions. However, in this paper we focus on matrices $\boldsymbol{D}$ of the form

$$\boldsymbol{D}=\left[\begin{array}{cc}\mathbf{0}_{(p+1)\times(p+1)} & \mathbf{0}_{(p+1)\times K}\\ \mathbf{0}_{K\times(p+1)} & \boldsymbol{\Sigma}^{-1}\end{array}\right],$$

where $\boldsymbol{\Sigma}$ is a positive definite matrix and $\mathbf{0}_{m\times l}$ is an $m\times l$ matrix of zeros. This matrix $\boldsymbol{D}$ penalizes the coefficients of the spline basis functions $(x-\kappa_k)_+^p$ only and will be used in the remainder of the paper. A standard choice is $\boldsymbol{\Sigma}=\boldsymbol{I}_K$.

Let $\boldsymbol{Y}=(y_1,y_2,\ldots,y_n)^T$, $\boldsymbol{X}$ be the matrix with the ith row $\boldsymbol{X}_i=(1,x_i,\ldots,x_i^p)$, and $\boldsymbol{Z}$ be the matrix with ith row $\boldsymbol{Z}_i=\left\{(x_i-\kappa_1)_+^p,\ldots,(x_i-\kappa_K)_+^p\right\}$. If we divide (1) by the error variance one obtains

$$\frac{1}{\sigma_\epsilon^2}\left\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{Z}\boldsymbol{b}\right\|^2+\frac{1}{\lambda\sigma_\epsilon^2}\boldsymbol{b}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{b}\ ,$$

where $\boldsymbol{\beta}=(\beta_0,\ldots,\beta_p)^T$ and $\boldsymbol{b}=(b_1,\ldots,b_K)^T$. Define $\sigma_b^2=\lambda\sigma_\epsilon^2$, consider the vector $\boldsymbol{\beta}$ as fixed parameters and the vector $\boldsymbol{b}$ as a set of random parameters with $E(\boldsymbol{b})=0$ and $\mathrm{cov}(\boldsymbol{b})=\sigma_b^2\boldsymbol{\Sigma}$. If $(\boldsymbol{b}^T,\boldsymbol{\epsilon}^T)^T$ is a normal random vector and $\boldsymbol{b}$ and $\boldsymbol{\epsilon}$ are independent

then one obtains an equivalent model representation of the penalized spline in the form of a LMM (Brumback et al., 1999). Specifically, the P-spline is equal to the best linear predictor (BLUP) in the LMM

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{b} + \boldsymbol{\epsilon}, \;\; \text{cov}\begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{\epsilon} \end{pmatrix} = \begin{bmatrix} \sigma_b^2\boldsymbol{\Sigma} & 0 \\ 0 & \sigma_\epsilon^2\boldsymbol{I}_n \end{bmatrix}. \tag{2}$$

For this model $E(\boldsymbol{Y}) = \boldsymbol{X}\boldsymbol{\beta}$ and $\text{cov}(\boldsymbol{Y}) = \sigma_\epsilon^2\boldsymbol{V}_\lambda$, where $\boldsymbol{V}_\lambda = \boldsymbol{I}_n + \lambda\boldsymbol{Z}\boldsymbol{\Sigma}\boldsymbol{Z}^T$ and n is the total number of observations. In the LMM described in (2) $\boldsymbol{X}$ corresponds to fixed effects.

3 The Canadian Age–Income Data

Figure 1 is a scatterplot of age versus log(income) for a sample of $n = 205$ Canadian workers, all of whom were educated to grade 13. These data were used by Ullah [26], who identifies their source as a 1971 Canadian Census Public Use Tape. Ruppert, Wand and Carroll [22] used a MATLAB program for the nonparametric Bayesian analysis of these data (See their Section 16.3).

3.1 Penalized spline model

We can model nonparametrically the conditional mean of log(income) as a function of age using penalized splines. We use a quadratic spline with $K = 15$ knots chosen so that the k-th knot is the sample quantile of age corresponding to probability $k/(K+1)$. We used the following model

$$\begin{cases} y_i &= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \sum_{k=1}^K b_k (x_i - \kappa_k)_+^2 + \epsilon_i \\ b_k &\sim N(0, \sigma_b^2) \\ \epsilon_i &\sim N(0, \sigma_\epsilon^2) \end{cases}, \tag{3}$$

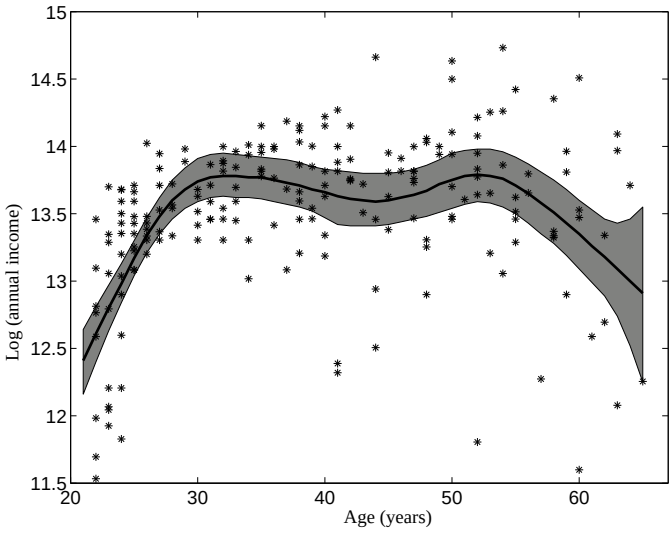
where y_i, x_i denote the log income and age of the i-th worker and b_k's and ϵ_i's are all assumed independent. To completely specify the Bayesian nonparametric model one needs to provide prior distributions for all model parameters. The following priors were used

$$\begin{cases} \beta_0,\, \beta_1,\, \beta_2 &\sim N(0, 10^6) \\ \sigma_b^{-2},\, \sigma_\epsilon^{-2} &\sim \text{Gamma}(0.001, 0.001) \end{cases}, \tag{4}$$

where the second parameter of the normal distribution is the variance. In many applications a normal prior distribution centered at zero with a standard error equal to 1000

is sufficiently noninformative. If there are reasons to suspect, either using alternative estimation methods or prior knowledge, that the true parameter is in another region of the space, then the prior should be adjusted accordingly. The parameterization of the Gamma(a, b) distribution is chosen so that its mean is $a/b = 1$ and its variance is $a/b^2 = 1,000$. Two other possible prior distributions for variance components are the uniform prior on $(0, M]$, and the uniform prior on $[-M, M]$ for the log of σ^2, where M is generally very large. We think that the practitioner should try a few dispersed priors in a sensitivity analysis to the choice of priors.

Figure 1: Scatterplot of log(income) versus age for a sample of $n = 205$ Canadian workers with posterior median (solid) and 95% credible intervals for the mean regression function



3.2 WinBUGS program for age–income data

We now describe the WinBUGS program that follows closely the description of the Bayesian nonparametric model in equations (3) and (4). We provide the entire program in Appendix A1. While this program was designed for the age–income data, it can be used for other penalized spline regression models with just minor adjustments. Many features of this program will be repeated in the other examples in this paper and changes will be described, as needed.

Equation (3) is the likelihood part of the model. The first and third lines of this equation are represented in the WinBUGS language as

```
for (i in 1:n)
    {response[i]~dnorm(m[i],taueps)
     m[i]<-inprod(beta[],X[i,])+inprod(b[],Z[i,])}
```

The number of subjects, n, is a constant in the program. The first statement specifies that the i-th response (log income of the i-th worker) has a normal distribution with mean m_i and precision $\tau_\epsilon = \sigma_\epsilon^{-2}$. The second statement provides the structure of the conditional mean function, $m_i = m(x_i)$. The first part of its definition contains the fixed effects while the second part contains the random effects. Here `beta[]` denotes the 1×3 dimensional vector $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$, which is the vector of fixed effects parameters. The ith row of matrix $\boldsymbol{X}$ is $\boldsymbol{X}_i = (1, x_i, x_i^2)$. The function `inprod` denotes the inner product of two vectors and `inprod(beta[],X[i,])` specifies the polynomial part of the regression in equation (3). Similarly, the vector $\boldsymbol{b}$ is the vector of coefficients of truncated polynomials and the ith row of matrix $\boldsymbol{Z}$ is the vector of truncated polynomials corresponding to the i-th observation

$$\boldsymbol{Z}_i = \left[(x_i - \kappa_1)_+^2, \ldots, (x_i - \kappa_K)_+^2\right] .$$

and `inprod(b[],Z[i,])` is the truncated polynomial part of the regression in equation (3).

The second line of equation (3) is coded as

```
for (k in 1:nknots){b[k]~dnorm(0,taub)}
```

which specifies that the coefficients b_k are independent and normally distributed with mean 0 and precision $\tau_b = \sigma_b^{-2}$. Here `nknots` is the number of knots ($K = 15$) and is introduced in WinBUGS as a constant.

The prior distributions of model parameters described in equation (4) are specified in WinBUGS as follows

```
for (l in 1:degree+1){beta[l]~dnorm(0,1.0E-6)}
taueps~dgamma(1.0E-3,1.0E-3)
taub~dgamma(1.0E-3,1.0E-3)
```

where `degree` denotes the degree of the spline ($p = 2$) and is introduced in the program as a constant. The prior normal distributions for the $\boldsymbol{\beta}$ parameters are expressed in terms of the precision parameter and the Gamma distributions are specified for the precision parameters $\tau_\epsilon = \sigma_\epsilon^{-2}$ and $\tau_b = \sigma_b^{-2}$.

Both matrices $\boldsymbol{X}$ and $\boldsymbol{Z}$ could be introduced as constants in WinBUGS. However we prefer to input only the covariates, the knots and the degree of the spline and obtain these matrices in WinBUGS. The following lines of code show how to obtain the matrix of fixed effects in WinBUGS

```
for (i in 1:n)
    {for (l in 1:degree+1){X[i,l]<-pow(covariate[i],l-1)}}
```

Here the function `pow` is the power function and `pow(a,b)` is a^b. The (i, l)-th entry of matrix $\boldsymbol{X}$ is thus specified to be x_i^{l-1} for $l = 1, \ldots, p+1$. Similarly, the matrix of random effects can be specified as follows

```
for (i in 1:n)
    {for (k in 1:K)
        {u[i,k]<-(covariate[i]-knot[k])*step(covariate[i]-knot[k])
         Z[i,k]<-pow(u[i,k],degree)}}
```

We introduced the auxiliary variable $u_{ik} = (x_i - \kappa_k)_+$ to make the code more readable. The truncation is obtained using the function `step`, where `step(x)` is 1 if x is positive and 0 otherwise. The (i, k)-th entry of the matrix $\boldsymbol{Z}$ is simply $z_{ik} = u_{ik}^p = (x_i - \kappa_k)_+^p$.

Note that the code is very short and intuitive presenting the model specification in rational steps. This is the main strength of WinBUGS coding and the expert or the practitioner would fully appreciate this after a few model implementations.

After writing the program one needs to load the data: the two n–dimensional vectors containing the response (y) and the covariate (x) values, the vector of knots, the sample size (n), the degree of the spline (p) and the number of knots (K). At this stage the program needs to be compiled and initial values for all random variables have to be loaded. One needs to make sure that all initial values are compatible with the model. For example, we need to choose strictly positive initial values for the precision parameters.

3.3 Model inference

Convergence to the posterior distributions was assessed by using several initial values of model parameters and visually inspecting several chains corresponding to the model parameters in Table 1. Convergence was attained in less than $1,000$ simulations, but we discarded the first $2,500$ burn-in simulations. For inference we used $25,000$ simulations and we monitored all parameters of the model. These simulations took approximately 13 minutes on a PC (3.6GB RAM, 3.4GHz CPU).

Table 1: Posterior median and 95% credible interval for some parameters of the model presented in equations (3) and (4)

Parameter	2.5%	50%	97.5%
β_0	4.18	5.82	8.49
β_1	0.21	0.43	0.51
β_2	-0.007	-0.005	-0.001
σ_b	0.011	0.017	0.028
σ_ϵ	0.49	0.54	0.59

Table 1 shows the posterior median and a 95% credible interval for some of the model parameters. We also obtained the posterior distributions of the mean function of the response, $m_i = m(x_i)$. Figure 1 displays the median, 2.5% and 97.5% quantiles of these posterior distributions for each value of the covariate x_i. The greyed area corresponds to credible intervals for each $m(x_i)$ and is not a joint credible band for the mean function. An important advantage of Bayesian over the typical frequentist analysis is that in the Bayesian case the credible intervals take into account the variability of each parameter and do not use the "plug-in" method. Prediction intervals at an in-sample x value can be obtained very easily by monitoring random variables of the type

$$y_i^* = m_i + \epsilon_i^* \, ,$$

with ϵ_i^* being independent realizations of the distribution $N(0, \sigma_\epsilon^2)$. This can be implemented by adding the following lines to the WinBUGS code

```
for (i in 1:n)
```

```
{epsilonstar[i]~dnorm(0,taueps)
 ystar[i]<-m[i]+epsilonstar[i]}
```

4 The Wage–Union Membership Data

Figure 2 displays data on wages and union membership for 534 workers described by Berndt [1]. The data were taken from the Statlib website at Carnegie Mellon University `lib.stat.cmu.edu/`. This data set was analyzed by Ruppert, Wand and Carroll [22] who show that standard linear, quadratic and cubic logistic regression are not appropriate in this case. Instead, they model the logit of the union membership probability as a penalized spline, which allows identification of features that are not captured by standard regression techniques. In this section we use their modeling approach and show how to quickly implement their model in WinBUGS.

4.1 Generalized Penalized spline model

Denote by y the binary union membership variable, by x the continuous wage variable and by $p(x)$ the union membership probability for a worker with wage x in dollars/hour. The logit of $p(x)$ is modeled nonparametrically using a linear ($p = 1$) penalized spline with $K = 20$ knots. We used the following model

$$\left\{ \begin{array}{lcl} y_i|x_i & \sim & \text{Bernoulli}\{p(x_i)\} \\ \text{logit}\{p(x_i)\} & = & \beta_0 + \beta_1 x_i + \sum_{k=1}^{K} b_k (x_i - \kappa_k)_+ \\ b_k & \sim & N(0, \sigma_b^2) \end{array} \right. , \tag{5}$$

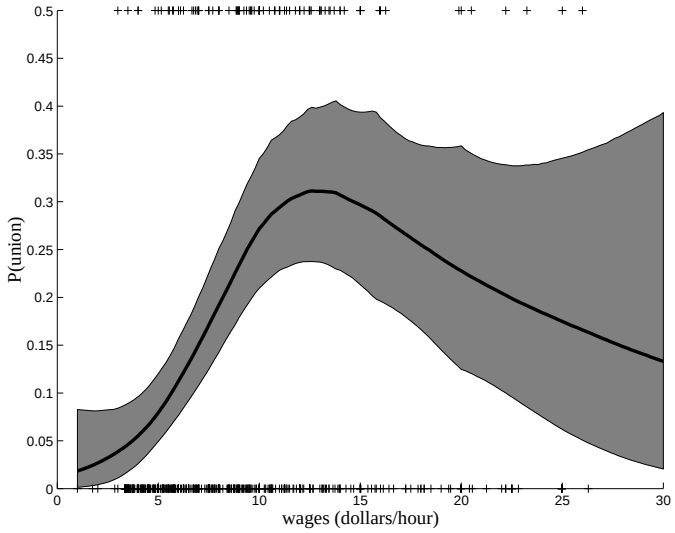
The following prior distributions were used

$$\left\{ \begin{array}{lcl} \beta_0,\ \beta_1 & \sim & N(0, 10^6) \\ \sigma_b^{-2} & \sim & \text{Gamma}(0.001, 0.001) \end{array} \right. . \tag{6}$$

4.2 WinBUGS program for wage–union data

While model (5) is very similar to model (3) the Bayesian analysis implementation in MATLAB, C or other software is significantly different. Typically, when the model is changed one needs to rewrite the entire code and make sure that all code bugs have been removed. This is a lengthy process that usually requires a high level of expertise in statistics and MCMC coding. WinBUGS cuts short this difficult process, thus making Bayesian analysis appealing to a larger audience.

Figure 2: Logistic spline fit to the union and wages scatterplot (solid) with 95% credible sets. Raw data are plotted as pluses, but with values of 1 for union replaced by 0.5 for graphical purposes. A worker making $44.50/hour was used in the fitting but not shown to increase detail.



In this case, changing the model from (3) to (5) requires only small changes in the WinBUGS code. Specifically, the two lines that specify the conditional distribution of the response variable are replaced by

```
for (i in 1:n)
    {response[i]~dbern(p[i])
     logit(p[i])<-inprod(beta[],X[i,])+inprod(b[],Z[i,])}
```

while the rest of the code remains practically unchanged. Given this very simple change, we do not provide the complete code for this model in the paper, but we provide a commented version in the accompanying software file.

4.3 Model inference

Table 2 shows the posterior median and a 95% credible interval for some of the model parameters. We also obtained the posterior distributions of $p_i = p(x_i)$ and Figure 2 displays the median, 2.5% and 97.5% quantiles of these distributions. The greyed area

corresponds to credible intervals for each $p(x_i)$ and is not a joint credible band. The credible intervals take into account the variability of each parameter.

Table 2: Posterior median and 95% credible interval for some parameters of the model presented in equations (5) and (6)

Parameter	2.5%	50%	97.5%
β_0	-7.48	-4.15	-2.43
β_1	-0.03	0.34	1.08
σ_b	0.045	0.100	0.229

5 The Coronary sinus potassium data

We consider the coronary sinus potassium concentration data measured on 36 dogs published by Grizzle and Allan [15] and Wang [30]. The measurements on each dog were taken every 2 minutes from 1 to 13 minute (7 observations per dog). The 36 dogs come from 4 treatments.

Wang [30] presents four smoothing spline analyses of variance models for this data. Crainiceanu and Ruppert [4] also present a hierarchical model of curves including a nonparametric overall mean, nonparametric treatment deviations from the overall curve, and nonparametric subject deviations from the treatment curves. In this section we show how to implement such a complex model in WinBUGS.

5.1 Longitudinal Nonparametric ANOVA model

Denote by y_{ij} and t_{ij} the potassium concentration and time for dog i at time j (in this example $t_{ij} = j$, but we keep the presentation more general). Consider the following model for potassium concentration

$$y_{ij} = f(t_{ij}) + f_{g(i)}(t_{ij}) + f_i(t_{ij}) + \epsilon_{ij} \ , \tag{7}$$

where $f(\cdot)$ is the overall curve, $f_{g(i)}(\cdot)$ are the deviations of the treatment group from the overall curve and $f_{ij}(\cdot)$ are the deviations of the subject curves from the group curves. Here $g(i)$ represents the treatment group index corresponding to subject i. All

three functions are modeled as linear splines. The curves are modeled as

$$\left\{\begin{array}{lcl} f(t) & = & \beta_0 + \beta_1 t + \sum_{k=1}^{K_1} b_k (t-\kappa_{1k})_+ \\ f_g(t) & = & \gamma_{0g} I_{(g>1)} + \gamma_{1g} t I_{(g>1)} + \sum_{k=1}^{K_2} c_{gk} (t-\kappa_{2k})_+ \\ f_i(t) & = & \delta_{0i} + \delta_{1i} t + \sum_{k=1}^{K_3} d_{ik} (t-\kappa_{3k})_+ \end{array}\right. \tag{8}$$

where $I_{(g>1)}$ is the indicator that $g > 1$, that is that the treatment group is $g = 2$, or 3 or 4. This specification of the treatment group curves is equivalent to

$$\left\{\begin{array}{lcl} f_1(t) & = & \sum_{k=1}^{K_2} c_{1k} (t-\kappa_{2k})_+ \\ f_g(t) & = & \gamma_{0g} + \gamma_{1g} t + \sum_{k=1}^{K_2} c_{gk} (t-\kappa_{2k})_+ \ , \ g = 2,\ 3,\ 4 \end{array}\right. \tag{9}$$

which avoids nonidentifiability of the slope and intercept parameters. The number of knots can be different and one can choose, for example, more knots to model the overall curve than each subject specific curve. However, in our example we used the same number of knots $K_1 = K_2 = K_3 = 3$.

The model also assumes that the b, c, d and δ parameters are mutually independent and

$$\left\{\begin{array}{lcl} b_k & \sim & N(0, \sigma_b^2),\ k = 1, \ldots, K_1 \\ c_{gk} & \sim & N(0, \sigma_c^2),\ g = 1, \ldots, 4,\ k = 1, \ldots, K_2 \\ d_{ik} & \sim & N(0, \sigma_d^2),\ i = 1, \ldots, N,\ k = 1, \ldots, K_3 \\ \delta_{0i} & \sim & N(0, \sigma_0^2),\ i = 1, \ldots, N \\ \delta_{1i} & \sim & N(0, \sigma_1^2),\ i = 1, \ldots, N \end{array}\right. , \tag{10}$$

where σ_b^2, σ_c^2 and σ_d^2 control the amount of shrinkage of the overall, group and individual curves respectively and σ_0^2 and σ_1^2 are the variance components of the subject random intercepts and slopes. We could also add other covariates that enter the model parametrically or nonparametrically, consider different shrinkage parameters for each treatment group, etc. All these model transformations can be done very easily in WinBUGS.

To completely specify the Bayesian nonparametric model one needs to specify prior distributions for all model parameters. The following priors were used

$$\left\{\begin{array}{lcl} \beta_0,\ \beta_1,\ \gamma_{0g}, \gamma_{1g} & \sim & N(0, 10^6),\ \ g = 1, \ldots, 4 \\ \sigma_b^{-2},\ \sigma_c^{-2}, \sigma_d^{-2},\ \sigma_\epsilon^{-2}, \sigma_0^{-2},\ \sigma_1^{-2} & \sim & \text{Gamma}(0.001, 0.001) \end{array}\right. . \tag{11}$$

5.2 WinBUGS program for the dog data

We provide the entire WinBUGS code for this model in Appendix A3. Equations (8) and (9) are coded in WinBUGS as

```
for (k in 1:n)
    {response[k]~dnorm(m[k],taueps)
     m[k] <-f[k]+fg[k]+fi[k]
     f[k] <-inprod(beta[],X[k,])+inprod(b[],Z[k,])
     fg[k]<-inprod(gamma[group[k],],X[k,])*step(group[k]-1.5)+
            inprod(c[group[k],],Z[k,])
     fi[k]<-inprod(delta[dog[k],],X[k,])+inprod(d[dog[k],],Z[k,])}
```

The response is organized as a column vector obtained by stacking the information for each dog. Because there are 7 observations for each dog, the observation number k can be written explicitly in terms of (i, j), that is $k = 7(i-1)+j$. The number of observations is $n = 36 \times 7 = 252$.

We used two $n \times 1$ column vectors with entries `dog[k]` and `group[k]`, that store the dog and treatment group indexes corresponding to the k-th observation.

The first two lines of code in the `for` loop correspond to equation (7), where `dnorm` specifies that `response[k]` has a normal distribution with mean `m[k]` and precision `taueps`. The mean of the response is specified to be the sum of `f[k]`, `fg[k]` and `fi[k]`, which are the variables for the overall mean, treatment group deviation from the mean and individual deviation from the group curves.

The last three lines of code in the `for` loop describe the structure of these curves in terms of splines. We keep the same notations from the previous sections, that is `X` stores the monomials and `Z` stores the truncated polynomials of the spline function. The WinBUGS code for the definition of `X` and `Z` is exactly the same as the one described in section 3.2 and is omitted here. Because in this example we use the same knots and covariates the matrices `X` and `Z` do not change for the three types of curves. However, it would be an useful exercise for the reader to write an alternative program using different matrices.

The definition of the overall curve `f[k]` follows exactly the same procedure with the one described in Section 3.2. The definition of `fg[k]` follows the same pattern but it involves two WinBUGS specific tricks. The first one is the use of the `step` function, described in Section 3.2. Here `step(group[k]-1.5)` is 1 if the index of the

group corresponding to the k-th observation is larger than 1.5 and zero otherwise. This captures the structure of the $f_g(\cdot)$ function in equation (9) because the possible values of `group[k]` are 1, 2, 3 and 4. The second trick is the nested indexing used in the definition of the $\boldsymbol{\gamma}$ and $\boldsymbol{c}$ parameters using the `dogs` vector described above. For example, the $\boldsymbol{\gamma}$ parameters are stored in a 4×2 matrix `gamma[,]` with the g-th line `gamma[g,]` corresponding to the parameters γ_{0g}, γ_{1g} of the $f_g(\cdot)$ function. Note that if g is replaced by `group[k]` we obtain the parameters corresponding to the k-th observation. Similarly, `c[,]` stores the c_{gk} parameters of $f_g(\cdot)$ and is a 4×3 matrix because there are 4 treatment groups and 3 knots. The definition of `fi[k]` curve uses the same ideas, with the only difference that the vector `dog[k]` is used instead of `group[k]`. Here, `delta[,]` is a 36×2 matrix with the i-th line containing the δ_{0i} and δ_{1i}, the random slope and intercept corresponding to the i-th dog. Also, `d[,]` is a 36×3 matrix with the i-th line storing the d_{i1}, d_{i2} and d_{i3}, the parameters of the truncated polynomial functions for the i-th dog.

The WinBUGS coding of the distributions of $\boldsymbol{b}$, $\boldsymbol{c}$, $\boldsymbol{d}$ and $\boldsymbol{\delta}$ follows almost literally the definitions provided in equation (10)

```
for (k in 1:nknots){b[k]~dnorm(0,taub)}
for (k in 1:nknots)
    {for (g in 1:ngroups){c[g,k]~dnorm(0,tauc)}}
for (i in 1:ndogs)
    {for (k in 1:nknots){d[i,k]~dnorm(0,taud)}}
for (i in 1:ndogs)
    {for (j in 1:degree+1){delta[i,j]~dnorm(0,taudelta[j])}}
```

For example, the parameters $c_{j,k}$ are assumed to be independent with distribution $N(0, \sigma_c^2)$ and the WinBUGS code is `c[g,k]~dnorm(0,tauc)`. Here `nknots`, `ngroups` and `ndogs` are the number of knots of the spline, the number of treatment groups and the number of dogs respectively. These are constants and are entered as data in the program.

Using the same notations as in Section 3.2 the normal prior distributions described in equation (11) are coded as

```
for (l in 1:degree+1){beta[l]~dnorm(0,1.0E-6)}
for (l in 1:degree+1)
    {for (j in 1:ngroups){gamma[j,l]~dnorm(0,1.0E-6)}}
```

and the prior gamma distributions on the precision parameters are coded as

```
taub~dgamma(1.0E-3,1.0E-3)
tauc~dgamma(1.0E-3,1.0E-3)
taud~dgamma(1.0E-3,1.0E-3)
taueps~dgamma(1.0E-3,1.0E-3)
for (j in 1:degree+1){taudelta[j]~dgamma(1.0E-3,1.0E-3)}
```

Here `taub`, `tauc`, `taud` and `taueps` are the precisions σ_b^{-2}, σ_c^{-2}, σ_d^{-2} and σ_ϵ^{-2} respectively. `degree` is the degree of the spline, which in our example is equal to 1 (linear spline) and `taudelta[1]` and `taudelta[2]` are the precisions σ_0^{-2} and σ_1^{-2} for the δ-parameters.

5.3 Model inference

Figure 3 shows the data for the 36 dogs corresponding to each treatment group together with the posterior mean and 90% credible interval for the treatment group mean functions. Recall that the treatment group functions are the sums between the overall mean function and the functions for the treatment group deviations from the mean functions, that is

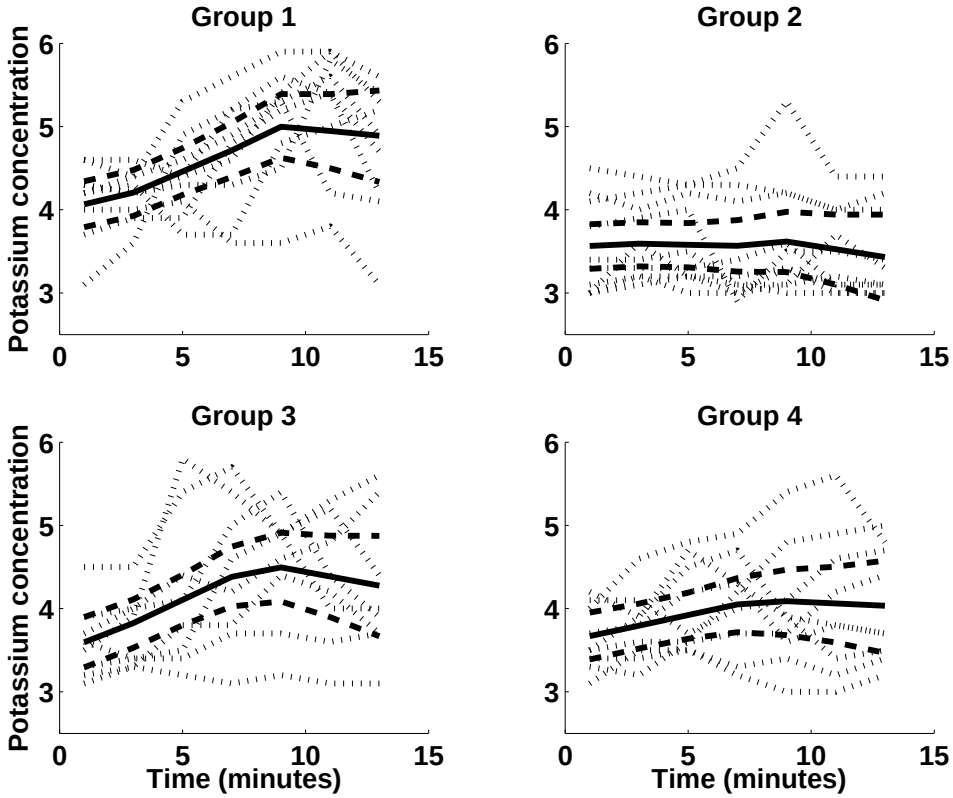
$$f_{\text{group}}(t) = f(t) + f_g(t)$$

This is achieved in WinBUGS by monitoring a new variable `fgroup[]` defined as

```
for (k in 1:n){fgroup[k]<-f[k]+fg[k]}
```

The total time for $2,500$ burn in and $25,000$ simulations from the target distribution took approximately 6.5 minutes (3.6Gb RAM, 3.4Ghz CPU).

Figure 3: Coronary sinus potassium concentrations for 36 dogs in four treatment groups with posterior median and 90% credible intervals of the group means



6 The ICR Recovery Data

Cryptosporidium parvum is a microscopic waterborne pathogen that can produce gastrointestinal illness in healthy individuals and serious complications or even death in individuals with a weakened immune system (see Meinhardt et al. [18]). In response to recent outbreaks, the US EPA conducted a national survey of *Cryptosporidium* concentrations under the Information Collection Rule (ICR) described by US EPA [8]. The results of the survey (pathogen counts) depend essentially on the random recovery rate, which is the probability that a lab technician observes and counts a pathogen included in the sample. To address this issue, the EPA conducted a parallel lab-spiking study, where laboratories analyzed samples with a known (only to the EPA) number of pathogens using the ICR procedure. For every experiment, collected data included volume of water filtered, number of organisms added to this volume, standard deviation of the number of organisms spiked, number of organisms counted by laboratories, volume of water analyzed by the laboratories, and three variables (turbidity, temperature and pH) measured at the time of filtration, and laboratory identification number. A total of $M = 21$ laboratories participated in the study providing 140 observations. For the purpose of this analysis we discarded three observations that did not contain all the covariates. See Scheller et al. [23] for a complete description of the data.

Crainiceanu et al. [6] describe a Bayesian parametric model that takes into account the various sources of variability in the data. In the following we generalize their approach by modeling nonparametrically the relationship between the logit of the recovery rates and some of the covariates.

6.1 Generalized additive mixed model

Let j be the j-th experiment at laboratory i, N_{ij} be the number of pathogens spiked in the total volume of water T_{ij}, and Z_{ij} be the observed count. The true concentration in the spiked sample is N_{ij}/T_{ij}. Let V_{ij} be the volume analyzed, W_{ij} be the turbidity,

L_{ij} be the temperature and M_{ij} be the pH of water. Consider the following model

$$\begin{cases} Z_{ij}|\lambda_{ij} & \sim & \text{Poisson}(\lambda_{ij}) \\ \lambda_{ij}|V_{ij}, N_{ij}, R_{ij}, T_{ij} & = & V_{ij}N_{ij}R_{ij}/T_{ij} \\ N_{ij}|a_{ij}, b_{ij} & \sim & \text{Gamma}(a_{ij}, b_{ij}) \\ \text{logit}(R_{ij})|V_{ij}, W_{ij}, L_{ij}, M_{ij}, \epsilon_{ij} & = & f_1(V_{ij}) + f_2(W_{ij}) + \theta_1 L_{ij} + \theta_2 M_{ij} + \epsilon_{ij} \\ \epsilon_{ij}|L_i, \sigma_\epsilon & \sim & N(L_i, \sigma_\epsilon^2) \\ L_i|\mu, \sigma_L & \sim & N(\mu, \sigma_L^2) \end{cases} \tag{12}$$

Conditional on the true concentration N_{ij}/T_{ij}, the expected number of pathogens in the volume analyzed V_{ij} is $V_{ij}N_{ij}/T_{ij}$. However, because of the lack of accuracy in the laboratory counting process, the expected number of pathogens is smaller. The expected number of pathogens actually counted has a Poisson distribution with mean $\lambda_{ij} = V_{ij}N_{ij}R_{ij}/T_{ij}$. Due to the nature of the spiking step the exact number N_{ij} is unknown but its mean and variance are known. Because $E[N_{ij}]$ is generally in the thousands, a continuous gamma distribution is satisfactory for modeling N_{ij}. The parameters a_{ij} and b_{ij} of the Gamma distribution in equation (12) are chosen so that

$$a_{ij}/b_{ij} = \text{E}[N_{ij}] \text{ and } a_{ij}/b_{ij}^2 = \text{Var}[N_{ij}] ,$$

where $\text{E}[N_{ij}]$ and $\text{Var}[N_{ij}]$ are known and read in as data.

The fourth relationship in the model (12) links the variability in the recovery rate to the volume analyzed, turbidity, temperature, pH of water and the random error component ϵ_{ij}. We model both $f_1(\cdot)$ and $f_2(\cdot)$ nonparametrically using quadratic penalized splines with $K = 10$ knots

$$\begin{cases} f_1(v) & = & \beta_{11}v + \beta_{12}v^2 + \sum_{k=1}^{K} b_{1k}(v - \kappa_{1k})_+^2 \\ f_2(w) & = & \beta_{21}v + \beta_{22}w^2 + \sum_{k=1}^{K} b_{2k}(w - \kappa_{2k})_+^2 \end{cases} \tag{13}$$

where b_{1k}, b_{2k} are independent for each $k = 1, \ldots, K$,

$$b_{1k} \sim N(0, \sigma_{1b}^2) , \quad b_{2k} \sim N(0, \sigma_{2b}^2) , \tag{14}$$

where σ_{1b}^2 and σ_{2b}^2 control the degree of smoothing of $f_1(\cdot)$ and $f_2(\cdot)$ respectively. Neither of the spline functions contains the intercept, because the overall mean μ is already contained in the definition of the laboratory effects.

The ϵ_{ij}'s are conditionally independent random errors with mean L_i representing random effects. Laboratory effects have normal distributions about the overall mean μ

with variance σ_L^2. To completely specify the Bayesian model one needs to specify prior distributions for all model parameters. The following priors were used

$$\begin{cases} \beta_{11},\ \beta_{12},\ \beta_{21},\ \beta_{22},\ \theta_1,\ \theta_2,\ \mu & \sim & N(0, 10^6),\ \ g = 1, \ldots, 4 \\ \sigma_{1b}^{-2},\ \sigma_{2b}^{-2}, \sigma_L^{-2},\ \sigma_\epsilon^{-2} & \sim & \text{Gamma}(0.001, 0.001) \end{cases} . \tag{15}$$

6.2 WinBUGS program for the ICR recovery model

We provide the entire WinBUGS code for this model in Appendix A4. The complexity of the recovery rate model is built by exploiting relatively simple conditional relationships. The WinBUGS code has the advantage that it almost literally reproduces these relationships. For example, the first seven relationships in equation (12) are represented as

```
for (k in 1:Nobservations)
    {Nobserved[k]~dpois(lambda[k])
     lambda[k]<-voanalyzed[k]*Nspiked[k]*R[k]/vototal[k]
     Nspiked[k]~dgamma(gammaa[k],gammab[k])
     logit(R[k])<-f1[k]+f2[k]+theta1*te[k]+theta2*ph[k]+epsilon[k]
     f1[k]<-inprod(beta1[],X1[k,])+inprod(b1[],Z1[k,])
     f2[k]<-inprod(beta2[],X2[k,])+inprod(b2[],Z2[k,])
     epsilon[k]~dnorm(L[lab[k]],tauepsilon)}
```

while the last is represented as

```
for (i in 1:M){L[i]~dnorm(mu,tauL)}
```

Here `Nobservations` denotes the total number of observations and there is a one to one onto relationship $(i, j) \leftrightarrow k$. `Nobserved[k]` is equal to Z_k and denotes the observed number of pathogens in the sample. `dpois(lambda[k])` specifies that Z_k has a Poisson distribution with parameter λ_k. The second line of code corresponds to the definition of λ_k, where `voanalyzed[k]` is the volume of water analyzed, `R[k]` is the random recovery rate and `vototal[k]` is the total volume of water spiked. `Nspiked[k]` is equal to N_k from equation 12 and denotes the number of pathogens spiked in the k-th sample. `gammaa[k], gammab[k]` are the a_k, b_k parameters of the Gamma distribution of N_k. These parameters are known and are introduced as constants in the program.

The following three lines of code specify the model of recovery rate, where `f1[k]` and `f2[k]` are the two nonparametric functions $f_1(v_k)$ and $f_2(w_k)$ respectively, and `epsilon[k]` is the random error component ϵ_k. Both nonparametric functions are modeled using penalized splines and coding follows a procedure similar to the one described for the simpler age-income example presented in Section 4.2. The only difference is that neither of the splines contains the intercept. The specification of the penalized splines is completed by

```
for (i in 1:nknots1){b1[i]~dnorm(0,taub1)}
for (i in 1:nknots2){b2[i]~dnorm(0,taub2)}
```

which specifies that the parameters for the truncated polynomials of functions $f_1(\cdot)$ and $f_2(\cdot)$ are independent with distributions $N(0, \sigma_{1b}^2)$ and $N(0, \sigma_{2b}^2)$, respectively. Note, that `taub1` and `taub1` denote the precisions of the normal distributions and not their variances. `nknots1` and `nknots2` denote the number of knots used for the two spline functions. The design matrices $\boldsymbol{X}_1$, $\boldsymbol{X}_2$ corresponding to fixed effects for $f_1(\cdot)$ and $f_2(\cdot)$ are introduced using the similar coding to the one presented in Section 4.2

```
for (k in 1:n)
    {for (l in 1:degree1){X1[k,l]<-pow(va[k],l)}
     for (l in 1:degree2){X2[k,l]<-pow(tu[k],l)}}
```

`degree1` and `degree2` are the degrees of the splines corresponding to $f_1(\cdot)$ and $f_2(\cdot)$ respectively. Here `va[]` and `tu[]` are the vectors containing the covariates volume analyzed and turbidity. The values of these two covariates are transformed, as described in the following section. The only difference from the definition of matrix `X` in Section 4.2 is that matrices `X1` and `X2` do not contain the column corresponding to the intercept. The definitions of the `Z1` and `Z2` follow exactly the same procedure with the one described in Section 4.2

```
for (i in 1:Nobservations)
    {for (k in 1:nknots1)
        {u1[i,k]<-(va[i]-knot1[k])*step(va[i]-knot1[k])
         Z1[i,k]<-pow(u1[k,i],degree1)}
```

```
for (k in 1:nknots2)
    {u2[i,k]<-(tu[i]-knot2[k])*step(tu[i]-knot2[k])
     Z2[i,k]<-pow(u2[i,k],degree2)}}
```

To specify the distribution of ϵ_k a standard nested indexing trick is used by introducing a new vector, `lam[k]`, which contains the laboratory identifiers. Therefore, the line of code

```
epsilon[k]~dnorm(L[lab[k]],tauepsilon)
```

is the WinBUGS representation of

$$\epsilon_{ij}|L_i, \sigma_\epsilon \sim N(L_i, \sigma_\epsilon^2) \; .$$

The distribution of L_i presented in the last line of equation (12) is coded as

```
for (i in 1:M){L[i]~dnorm(mu,tauL)}
```

Equation (14) is coded as

```
for (i in 1:nknots1){b1[i]~dnorm(0,taub1)}
for (i in 1:nknots2){b2[i]~dnorm(0,taub2)}
```

The prior normal distributions in equation (15) are specified as

```
for (l in 1:degree1){beta1[l]~dnorm(0,1.0E-6)}
for (l in 1:degree2){beta2[l]~dnorm(0,1.0E-6)}
mu~dnorm(0,1.0E-6); theta1~dnorm(0,1.0E-6); theta2~dnorm(0,1.0E-6)
```

and the prior Gamma distributions of the precision parameters are defined as

```
taub1~dgamma(1.0E-3,1.0E-3); taub2~dgamma(1.0E-3,1.0E-3)
tauL~dgamma(1.0E-3,1.0E-3); tauepsilon~dgamma(1.0E-3,1.0E-3)
```

6.3 Model inference

Convergence to the posterior distribution was attained in less than $2,000$ simulations, but we discarded the first $2,500$ burn-in simulations. For inference we used $25,000$ simulations and we monitored all parameters of the model. These simulations took approximately 4.5 minutes on a PC (3.6GB RAM, 3.4GHz CPU).

We used a few data transformations to improve MCMC mixing and numerical stability. Both covariates that entered the model linearly, temperature and pH, were centered and standardized. Volume analyzed used as a covariate was log transformed. Turbidity could not be log transformed because of one zero value. Instead we used the transformation

$$w \to \log\left(w + \bar{W}/10\right) \ ,$$

where $\bar{W}$ is the average of all turbidity values. After these transformations both the volume analyzed and turbidity entered the model via nonparametric functions using penalized splines. Using penalized splines with the original data should provide similar results. However, both covariates contain outliers, especially turbidity, and the mixing properties for some parameters were very poor. This problem was not experienced after data transformations. Another trick used for improving mixing was the hierarchical centering of random effects. In Section 7 we provide a detailed discussion of strategies for improving MCMC mixing.

Table 3 displays the posterior mean and 95% credible interval for some parameters of the model. The 95% credible interval for temperature does not contain zero, which shows that water temperature has a significant, if small, effect on the recovery rate. pH does not have a significant effect on recovery rate. To examine the importance of laboratory effects in explaining the variability in the data we computed the posterior distribution of

$$RS = \frac{\sigma_L^2}{\sigma_L^2 + \sigma_\epsilon^2} \ .$$

The RS statistic is inspired from standard linear regression and represents the fraction of the total variance explained by the random laboratory effects. RS is smaller than 0.085 with 0.95 probability and smaller than 0.1 with 0.975 probability, showing that laboratory effects explain only a very small fraction of the data variation.

Table 3: Posterior median and 95% credible interval for some model parameters

Parameter	2.5%	50%	97.5%
θ_1	-0.46	-0.24	-0.04
θ_2	-0.10	0.12	0.33
σ_{1b}	0.03	0.07	0.17
σ_{2b}	0.03	0.16	0.50
σ_L	0.03	0.10	0.39
σ_ϵ	0.92	1.08	1.29
RS	5×10^{-4}	0.009	0.116

Figure 6.3 shows the effect of the volume analyzed and turbidity on the logit of recovery rate by displaying the posterior medians and 95% credible intervals for the functions $f_1(v)$ and $f_2(w)$.

Figure 6.3–(a) shows a general decreasing pattern of the effect of the volume analyzed on the recovery rates, which is more pronounced for small values of log volume. However, it seems that for values of log volume larger than 2.5 this pattern disappears. The credible intervals are wider closer to the boundaries because less data corresponds to very low or large values of the volume of water analyzed. For example, 75% of the log volume data lie in the interval $[-1, 3]$ and only one datum is smaller than -3.

Figure 6.3–(b) shows that all 95% credible intervals for $f_2(w)$ contain zero for each in-sample value of the covariate. This suggests that turbidity does not have a significant effect on the recovery rates.

Simple, nonparametric estimates of recovery rates can be obtained by ignoring the Poisson variability of pathogen counts in equation (12), which gives the "empirical recovery rates"

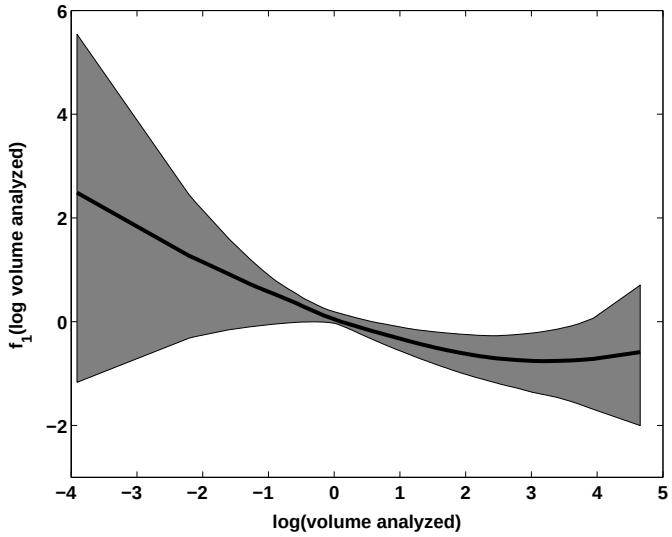
$$\widehat{R}_{ij} = \frac{Z_{ij}T_{ij}}{N_{ij}V_{ij}} \ .$$

Figure 5 displays these values relative to the log of the volume analyzed. To capture the individual effect of volume analyzed on recovery rates we obtained the posterior distribution of the recovery rate when all other covariates are fixed at their sample median value. For each value of log volume analyzed, v, we monitored $R_{0.5}(v)$ where

$$\text{logit}\left\{R_{0.5}(v)\right\} = f_1(v) + f_2(w_{0.5}) + \theta_1 L_{0.5} + \theta_2 M_{0.5} + \mu \ ,$$

Figure 4: (a) – Posterior median and 95% credible intervals for $f_1(v)$, the spline function describing the effect of log of volume analyzed on the logit of recovery rate. (b) – Posterior median and 95% credible intervals for $f_2(v)$, the spline function describing the effect of turbidity on the logit of recovery rate. The x–axis represents the transformed values of turbidity, $\log(w + \bar{W}/10)$.

(a)



(b)

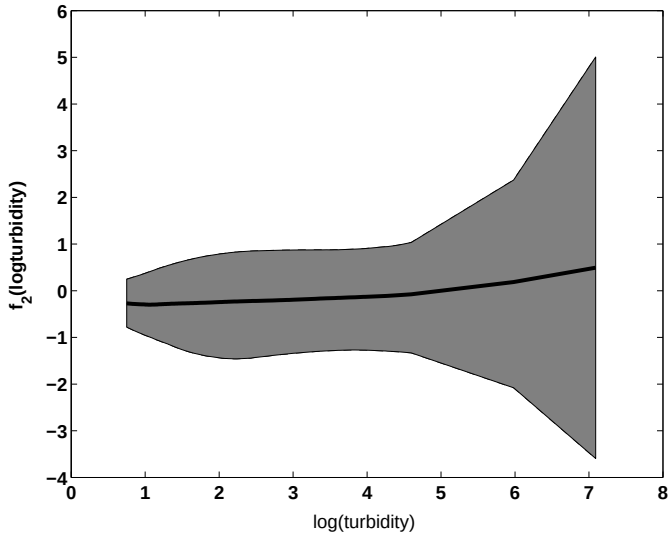
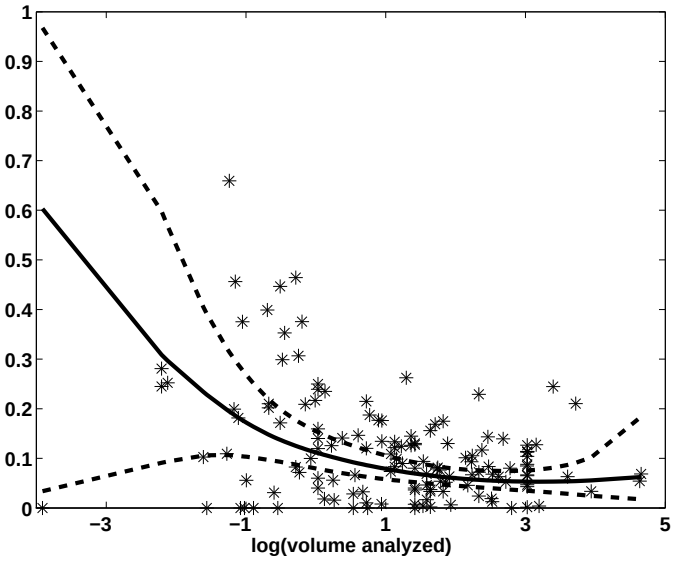


Figure 5: Log volume analyzed versus empirical rates. Posterior median (solid line) an 95% credible intervals (dashed lines) for the effect of volume analyzed on recovery rates



and $w_{0.5}$, $L_{0.5}$, $M_{0.5}$ are the sample medians of turbidity, temperature and pH respectively. As was previously discussed, laboratory effects did not capture a significant proportion of variability and were ignored in this formula. The corresponding WinBUGS code is

```
for (k in 1:Nobservations)
    {logit(R05[k])<-f1[k]+f2[110]+theta1*te[18]+theta2*ph[6]+mu}
```

Here 110, 18 and 6 are the indexes of the median observations for turbidity, temperature and pH respectively.

Figure 5 also shows the posterior median and 95% credible interval for $R_{0.5}(v)$ for all in-sample values of log volume analyzed. It is interesting that recovery rate is decreasing for log volume of water analyzed approximately between $[-4, 1]$ and seems to be relatively constant after 1. This type of pattern could not be captured by a logistic-linear regression.

7 Improving mixing

Mixing is the property of the Markov chain to move rapidly throughout the support of the posterior distribution of the parameters. Improving mixing is very important especially when computation speed is affected by the size of data set or model complexity. In this section we present a few simple but effective techniques that help improve mixing.

One of the most commonly used methods in regression is centering and standardizing the covariates, which make the covariates orthogonal to the intercept and improve numerical stability. In our experience with WinBUGS, these transformations also improve, sometimes dramatically, mixing properties of simulated chains.

Another, less known technique is hierarchical centering (Gelfand et al. [12]; Gelfand et al. [13]). Many statistical models contain random effects that are ordered in a natural hierarchy (e.g. observation/site/region). Consider the example in Section 6 where the logit of the recovery rate is modeled as

$$\begin{cases} \text{logit}(R_{ij}) & = & f_1(V_{ij}) + f_2(W_{ij}) + \theta_1 L_{ij} + \theta_2 M_{ij} + \epsilon_{ij} \\ \epsilon_{ij}|L_i, \sigma_s & \sim & N(L_i, \sigma_\epsilon^2) \\ L_i|\mu, \sigma_L & \sim & N(\mu, \sigma_L^2) \end{cases} . \qquad (16)$$

The hierarchy of random effects has two levels, where the mean of ϵ_{ij} is the laboratory effect L_i and the mean of L_i is the overall mean μ. This model specification, that we used in our WinBUGS coding, uses hierarchical centering. Note that model (16) has the equivalent, uncentered, form

$$\begin{cases} \text{logit}(R_{ij}) & = & f_1(V_{ij}) + f_2(W_{ij}) + \theta_1 L_{ij} + \theta_2 M_{ij} + \epsilon_{ij} + L_i + \mu \\ \epsilon_{ij}|L_i, \sigma_s & \sim & N(0, \sigma_\epsilon^2) \\ L_i|\mu, \sigma_L & \sim & N(0, \sigma_L^2) \end{cases} . \qquad (17)$$

The coding of this model in WinBUGS could be implemented by replacing the following code

```
for (i in 1:Nobservations)
     {lambda[k]<-voanalyzed[k]*Nspiked[k]*R[k]/vototal[k]
      logit(R[k])<-f1[k]+f2[k]+theta1*te[k]+theta2*ph[k]+epsilon[k]
      epsilon[k]~dnorm(L[lab[k]],tauepsilon)}
for (i in 1:M){L[i]~dnorm(mu,tauL)}
```

with

```
for (i in 1:Nobservations)
     {lambda[k]<-voanalyzed[k]*Nspiked[k]*R[k]/vototal[k]
      logit(R[k])<-f1[k]+f2[k]+theta1*te[k]+theta2*ph[k]+
                  epsilon[k]+L[lab[k]]+mu
      epsilon[k]~dnorm(0,tauepsilon)}
for (i in 1:M){L[i]~dnorm(0,tauL)}
```

The hierarchical centering of random effects generally has a positive effect on simulation mixing and we recommend it whenever the model contains a natural hierarchy. Bayesian smoothing models presented in this paper also contain the exchangeable random effects, $\boldsymbol{b}$, which are not part of an hierarchy and they cannot be "hierarchically centered".

Ideally, we would like to have good mixing for all parameters of the model, but in some cases (e.g. complex models for large data sets) mixing may be good for some parameters and very poor for others. For example, in the Bayesian smoothing examples presented in this paper the mixing properties of the mean function are usually better than those of truncated polynomial coefficients. This is most probably due to the fact that more information is available about the mean function than about individual parameters. Crainiceanu et al. [5] show that even for a simple Poisson–Log Normal model the amount of information has a strong impact on the mixing properties of parameters. A practical recommendation in these cases is to improve mixing, as much as possible, for a subset of parameters of interest. These model specification refinements pay off especially in slow WinBUGS simulations.

8 Script files in WinBUGS

The scripting language is an alternative to the menu/dialog box interface of WinBUGS. This language can be useful for automating routine analysis. The language replaces clicking on buttons by code commands.

A minimum of four files are required:

a. The script file itself. This file has to be in WinBUGS format (.odc)

b. A file containing the BUGS language representation of the model (.txt)

c. A file (or several) containing the data (.txt)

d. A file for each chain containing initial values (.txt).

The shortcut BackBUGS has been set up to run the commands contained in the file script.odc (in the root directory of WinBUGS) when it is double-clicked. A WinBUGS session may be embedded within any software component that can execute the BackBUGS shortcut.

A list of currently implemented commands in the scripting language can be found in the WinBUGS manual.

We provide an example of script file that we used for our penalized splines model described in Section 3. This file can be used with the `PsplineBayes.txt` file that contains the actual model description. See Appendix A1 for a complete description of the model. Data and initial values are contained in the file `ageincome_dat.txt` and `ageincome_in.txt` respectively.

```
display('log')                          #Choose display type
check('Test1/PsplineBayes.txt')         #Model check
data('Test1/ageincome_dat.txt')         #Load data
compile(1)                              #Compile model
inits(1, 'Test1/ageincome_in.txt')      #Initialize parameters
gen.inits()                             #Generate initial values
update(4000)                            #4000 burn-in iterations
set(m[])                                #Set m[] to be monitored
set(beta[])
set(sigmab)
set(sigmaepsilon)
update(100000)                          #Do 100000 iterations
stats(*)                                #Provide stats for all parameters
history(*)                              #Provide chain histories
trace(*)                                #Provide chain traces
```

```
density(*)                          #Provide kernel density estimates
autoC(*)                            #Provide autocorrelations
quantiles(*)                        #Provide running quantiles
coda(*,output)                      #Save the two files for coda
save('ageincomeLog')                #Save the output file
```

9 Pros, pros and cons

An important advantage of WinBUGS is the simple programming that translates almost literally the Bayesian model into code. This in turn saves time by avoiding the usually lengthy implementations of the MCMC simulation algorithms. For example, after deciding on the model for recovery rates presented in Section 6, we estimate that total programming time is approximately 1–2 hours. The same program implementation in C or MATLAB would probably take days and even more probably, weeks.

Programs designed by experts for specific problems can be more refined by taking into account specific properties of the model and using a combination of art and experience to improve mixing and computation time. However, when we compare a WinBUGS with an expert program in terms of computation speed, programming time needs to be taken into account. What matters in the end is time viewed as the sum of time spent on programming and simulations as well as accuracy of results. Moreover with increased computational capabilities it is likely that the simulation time will continue to decrease while coding time will remain practically the same.

WinBUGS allows simple model changes to be reflected in simple code changes, which encourages the practitioner or the expert to investigate a much wider spectrum of models. Expert programs are usually restrictive in this sense. Imagine, for example, that we would like to change the program recovery rate model presented in Section 6 by adding two new covariates and changing the distribution of random effects from a normal to a Student's t distribution. This takes minutes in WinBUGS and probably weeks for C implementation.

Our recommendation is to start with WinBUGS, implement the model for the specific data set. If it runs in a reasonable time, say less than 24 hours (most programs will run in minutes or hours), then continue to use WinBUGS. Otherwise consider

designing an expert program. Even if one decides to use the expert program we still recommend using WinBUGS as a method of checking results. Programming errors and debugging time are also dramatically reduced in WinBUGS.

There are, of course, limitations to what WinBUGS can do. For example, if the posterior distribution has two (or more) isolated modes then the MCMC chain will probably converge to the distribution around one of the modes and never jump to a neighborhood of the other mode. This problem could be overcome using, for example, simulated annealing or tempering, but WinBUGS does not yet have such capabilities. The multiple modes could be detected in WinBUGS using multiple starting points if one knew the number of modes and their approximate location. Another limitation is that WinBUGS does not have (or we do not know how to do) model averaging. These two features form our short "wish list" for future WinBUGS developments.

Appendix A1: WinBUGS code for the age-income example.

Model. This is the complete code for scatterplot smoothing used in the age-income example. It was stored in the PsplineBayes.txt file for use with the script file described in Section 8.

```
    model{                                  #Begin model
#This model can be used for any simple scatterplot smoothing. It
#can be easily modified to accommodate other covariates and/or
#random effects

#Likelihood of the model
    for (i in 1:n)
        {response[i]~dnorm(m[i],taueps)
         m[i]<-inprod(beta[],X[i,])+inprod(b[],Z[i,])}

#Prior distributions of the random effects parameters
    for (k in 1:nknots){b[k]~dnorm(0,taub)}

#Prior distribution of the fixed effects parameters
    for (l in 1:degree+1){beta[l]~dnorm(0,1.0E-6)}

#Prior distributions of the precision parameters
    taueps~dgamma(1.0E-3,1.0E-3); taub~dgamma(1.0E-3,1.0E-3)

#Construct the design matrix of fixed effects
    for (i in 1:n)
        {for (l in 1:degree+1){X[i,l]<-pow(covariate[i],l-1)}}

#Construct the design matrix of random effects
    for (i in 1:n)
        {for (k in 1:nknots)
```

```
            {u[i,k]<-(covariate[i]-knot[k])*step(covariate[i]-knot[k])
             Z[i,k]<-pow(u[i,k],degree)}}

#Deterministic transformations. Obtain the standard deviations and
#the smoothing parameter
    sigmaeps<-1/sqrt(taueps);sigmab<-1/sqrt(taub)
    lambda<-pow(sigmab,2)/pow(sigmaeps,2)

#Predicting new observations
    for (i in 1:n)
        {epsilonstar[i]~dnorm(0,taueps)
         ystar[i]<-m[i]+epsilonstar[i]}

    }                                   #end model
```

Data. Data are entered using the following two structures

```
covariate[]     response[]
21.0            11.1563
22.0            12.8131
22.0            13.0960
...             ...
63.0            14.0916
64.0            13.7102
65.0            12.2549
END
```

and

```
list(n=205,nknots=15,degree=2,
     knot=c(23,24,25,27,30,32,35,38,40,42,46,49,52,55,59))
```

The following notation was used:

1. `response[]`: $n \times 1$ vector containing the response variable y.
2. `covariate[]`: $n \times 1$ vector containing the covariate variable x.
3. `n`: sample size, n
4. `nknots`: number of knots, K, of the spline
5. `knot[]`: $K \times 1$ vector containing the knots of the spline
6. `degree`: degree of the spline, p

Initial values. Initial values are entered using the following structure. The rest of uninitialized values can be generated within WinBUGS

```
list(beta=c(0,0,0), b=c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),
     taueps=0.01, taub=0.01)
```

Parameters initialized are

1. `beta[]`: coefficients of the monomials
2. `b[]`: coefficients of the truncated basis functions
3. `taueps`: precision of the ϵ's
4. `taub`: precision of b's

Appendix A2: WinBUGS code for the age-income example.

Omitted here because it is very similar to age-income example. For a complete commented program see the model "P-spline fitting with Bernoulli variation" in the attached WinBUGS file.

Appendix A3: WinBUGS code for coronary sinus potassium example.

Model. This is the complete code for the Bayesian semiparametric model for coronary sinus potassium example presented in Section 5.

```
    model{                                  #Begin model
#This model was designed for the coronary sinus potassium model
#described in this paper. However, the basic coding ideas can be
#applied more generally to longitudinal models that involve a
#hierarchy of parametric and/or nonparametric curves

#Likelihood of the model
    for (k in 1:n)
        {response[k]~dnorm(m[k],taueps)
         m[k]<-f[k]+fg[k]+fi[k]
         f[k]<-inprod(beta[],X[k,])+inprod(b[],Z[k,])
         fg[k]<-inprod(gamma[group[k],],X[k,])*step(group[k]-1.5)+
               inprod(c[group[k],],Z[k,])
         fi[k]<-inprod(delta[dog[k],],X[k,])+inprod(d[dog[k],],Z[k,])}

#Prior for truncated polynomial parameters for the overall curve
    for (k in 1:nknots){b[k]~dnorm(0,taub)}

#Prior for truncated polynomial parameters for the curves
#describing group deviations from the overall curve
    for (k in 1:nknots)
        {for (g in 1:ngroups){c[g,k]~dnorm(0,tauc)}}

#Prior for truncated polynomial parameters for the individual
#deviations from the group curve
    for (i in 1:ndogs)
        {for (k in 1:nknots){d[i,k]~dnorm(0,taud)}}
```

```
#Prior for monomial parameters of the overall curve
    for (l in 1:degree+1){beta[l]~dnorm(0,1.0E-6)}

#Prior for monomial parameters of curves describing the group
#deviations from the overall curve
    for (l in 1:degree+1)
        {for (j in 1:ngroups){gamma[j,l]~dnorm(0,1.0E-6)}}

#Prior for monomial parameters of curves describing the individual
#deviations from the group curve
    for (i in 1:ndogs)
        {for (j in 1:degree+1){delta[i,j]~dnorm(0,taudelta[j])}}

#Priors of precision parameters
    taub~dgamma(1.0E-3,1.0E-3)
    tauc~dgamma(1.0E-3,1.0E-3)
    taud~dgamma(1.0E-3,1.0E-3)
    taueps~dgamma(1.0E-3,1.0E-3)
    for (j in 1:degree+1){taudelta[j]~dgamma(1.0E-3,1.0E-3)}

#Construct the design matrix of fixed effects
    for (i in 1:n)
        {for (l in 1:degree+1){X[i,l]<-pow(covariate[i],l-1)}}

#Construct the design matrix of random effects
    for (i in 1:n)
        {for (k in 1:nknots)
            {u[i,k]<-(covariate[i]-knot[k])*step(covariate[i]-knot[k])
             Z[i,k]<-pow(u[i,k],degree)}}
```

```
#Define the group curves
    for (i in 1:n){fgroup[i]<-f[i]+fg[i]}


        }                                   #End model
```

Data. Data is contained into two structures, the first one containing data in rectangular format and the second in a list format.

```
response[]  dog[]   group[] covariate[]
    4       1       1       1
    4       1       1       3
    ...     ...     ...     ...
    4.9     36      4       11
    5       36      4       13
    END
```

Data in scalar or vector format is introduced using the list environment

```
list(list(n=252,nknots=3,degree=1,ngroups=4,ndogs=36
     knot=c(3,7,9))
```

The following notations were used:

1. `response[]`: vector of potassium concentrations.
2. `dog[]`: vector of dog indexes.
3. `group[]`: vector of group indexes.
4. `covariate`: vector covariates representing time (minutes).
5. `n`: total number of observations.
6. `nknots`: numbers of knots used in the spline curves.
7. `degree`: degree of the splines.
8. `ngroups`: number of treatment groups
9. `ndogs`: number of dogs

10. `knot[]`: vector of knots used in the splines

Initial values After loading the data, the model is compiled and parameters need to be initialized. We used the following structures for initial values.

For monomial coefficients of curves describing the group's deviations from the overall curve

```
gamma[,1]   gamma [,2]
    0       0
    0       0
    0       0
    0       0
    END
```

For truncated polynomial coefficients of curves describing the group's deviations from the overall curve

```
    c[,1]   c[,2]   c[,3]
    0       0       0
    0       0       0
    0       0       0
    0       0       0
    END
```

For monomial coefficients of curves describing the individual deviations from the group curves

```
    delta[,1]   delta[,2]
    0           0
    0           0
    ...         ...
    0           0
    0           0
    END
```

For truncated polynomial coefficients of curves describing the individual deviations from the group curves

```
d[,1]   d[,2]   d[,3]
0       0       0
0       0       0
...     ...     ...
0       0       0
0       0       0
END
```

We also initialized the following parameters

```
list(beta=c(0,0),b=c(0,0,0),taueps=0.01,taub=0.01,
     tauc=0.01,taud=0.01,taudelta=c(0.1,0.1))
```

and generated initial values for all the other parameters within WinBUGS. The following notations were used

1. `beta[]`: vector containing the monomial coefficients of the overall mean curve.
2. `b[]`: vector containing the truncated polynomial coefficients of the overall mean curve.
3. `taueps`: precision of the error distribution.
4. `taub`: precision of truncated polynomial coefficients for the overall curve.
5. `tauc`: precision of truncated polynomial coefficients for the spline curve representing the group deviations from the overall curve.
6. `taud`: precision of truncated polynomial coefficients for the spline curve representing the individual deviations from the group curve.
7. `taudelta[]`: vector of precisions of monomial coefficients for the spline curve representing the individual deviations from the group curve.

Appendix A4: WinBUGS code for the recovery rate example.

Model. This is the complete code for the Bayesian semiparametric model for pathogen recovery rate presented in Section 6.

```
    model{                                  #Begin model
#This model was designed for the recovery rate model described in
#this paper. However, the basic coding ideas can be used in many
#GLMM models where one or more covariates enter the model
#nonparametrically

#Model structure
    for (k in 1:Nobservations)
        {Nobserved[k]~dpois(lambda[k])
         lambda[k]<-voanalyzed[k]*Nspiked[k]*R[k]/vototal[k]
         Nspiked[k]~dgamma(gammaa[k],gammab[k])
         logit(R[k])<-f1[k]+f2[k]+theta1*te[k]+theta2*ph[k]+epsilon[k]
         f1[k]<-inprod(beta1[],X1[k,])+inprod(b1[],Z1[k,])
         f2[k]<-inprod(beta2[],X2[k,])+inprod(b2[],Z2[k,])
         epsilon[k]~dnorm(L[lab[k]],tauepsilon)}

    for (i in 1:M){L[i]~dnorm(mu,tauL)}

#Distribution of the truncated polynomial coefficients
    for (i in 1:nknots1){b1[i]~dnorm(0,taub1)}
    for (i in 1:nknots2){b2[i]~dnorm(0,taub2)}

#Prior distributions for monomial coefficients
    for (l in 1:degree1){beta1[l]~dnorm(0,1.0E-6)}
    for (l in 1:degree2){beta2[l]~dnorm(0,1.0E-6)}

#Prior distribution for overall mean
```

```
    mu~dnorm(0,1.0E-6)

#Prior distributions for temperature and pH parameters
    theta1~dnorm(0,1.0E-6)
    theta2~dnorm(0,1.0E-6)

#Prior distributions for precision parameters
    taub1~dgamma(1.0E-3,1.0E-3)
    taub2~dgamma(1.0E-3,1.0E-3)
    tauL~dgamma(1.0E-3,1.0E-3)
    tauepsilon~dgamma(1.0E-3,1.0E-3)

#Design matrices for monomials of the spline functions
    for (i in 1:Nobservations)
        {for (l in 1:degree1){X1[i,l]<-pow(va[i],l)}
         for (l in 1:degree2){X2[i,l]<-pow(tu[i],l)}}

#Design matrices for truncated polynomials
    for (i in 1:Nobservations)
        {for (k in 1:nknots1)
            {u1[i,k]<-(va[i]-knot1[k])*step(va[i]-knot1[k])
             Z1[i,k]<-pow(u1[i,k],degree1)}
        for (k in 1:nknots2)
            {u2[i,k]<-(tu[i]-knot2[k])*step(tu[i]-knot2[k])
             Z2[i,k]<-pow(u2[i,k],degree2)}}

#Deterministic transformations
    sigmaepsilon<-1/sqrt(tauepsilon)
    sigmaL<-1/sqrt(tauL)

#Fraction of total variability explained by laboratory effects
```

```
    RS<-pow(sigmaL,2)/(pow(sigmaL,2)+pow(sigmaepsilon,2))

#Effect of volume of water analyzed on turbidity rates when all
#the other covariates are set equal to their median value. Lab
#effects are ignored
    for (k in 1:Nobservations)
        {logit(R05[k])<-f1[k]+f2[110]+theta1*te[18]+theta2*ph[6]+mu}

    }                                   #end model
```

Data. Data is contained into two structures, the first one containing data in rectangular format and the second in a list format. For presentation purposes we break down the rectangular format data into two parts. These two parts could be included in the program separate or together and are presented below

```
gammaa[]    gammab[]    lab[]   voanalyzed[]    va[]    vototal[]
51.24       0.0107      15      5.97            1.79    102.95
40.80       0.0064      16      12.53           2.53    111.85 ...
...         ...     ...             ...     ... 78.90       0.0079
17      20.59           3.02    102.95 19.40       0.0019      1
0.75            -0.29  102.95 END

Nobserved[]     tu[]    te[]    ph[] 15             2.02    0.589
0.727 49            1.23    -0.402  -0.293 ...              ...
...     ... 253             0.92    -0.550  0.489 6
4.60    -2.192  -1.142 END
```

Data in scalar or vector format is introduced using the list environment

```
list(Nobservations=137,M=21,degree1=2,degree2=2,nknots1=10,nknots2=10,
     knot1=c(-1.0498,-0.2877,0.0296, 0.7227,1.1282,1.4159,1.7733,
             2.3321, 2.8015, 3.1523),
     knot2=c(1.0143,1.1044,1.2289,1.3395,1.4463, 1.6896, 2.0940,
             2.4499, 3.0016, 3.9725))
```

The following notations were used:

1. `gammaa[]`, `gammab[]`: vectors of parameters for the Gamma distributions of the number of pathogens in the volume of water.
2. `lab[]`: vector of lab indexes.
3. `voanalyzed[]`: vector of volumes of water analyzed
4. `va[]`: vector of log of volumes analyzed
5. `vototal[]`: vector of total volumes used for spiking
6. `Nobserved[]`: vector of number of pathogens counted in the volume of water analyzed
7. `tu[]`: vector of transformed values of turbidity (see Section 6.3).
8. `te[]`, `ph[]`: vectors of centered and standardized temperature and pH of water
9. `Nobservations`: number of observations
10. `M`: number of laboratories
11. `degree1`, `degree2`: The degrees of the splines used to model volume analyzed and turbidity
12. `nknots1`, `nknots2`: The number of knots of the splines used to model volume analyzed and turbidity
13. `knot1[]`, `knot2[]`: The vectors of knots for the splines used to model volume analyzed and turbidity

Initial values After loading the data, the model is compiled and parameters need to be initialized. We used the following two structures for initial values

```
epsilon[]   Nspiked[]
    0       0.01
    0       0.01
    ...     ...
    0       0.01
    0       0.01
    END
```

and

```
list(mu=0,theta1=0,theta2=0,taub1=0.01,taub2=0.01,tauL=0.01,
     tauepsilon=0.01,beta1=c(0,0),beta2=c(0,0),
     b1=c(0,0,0,0,0,0,0,0,0,0),b2=c(0,0,0,0,0,0,0,0,0,0),
     L=c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0))
```

The following notations were used

1. `epsilon[]`: vector of errors that appears in the expression of logit recovery rate
2. `Nspiked[]`: vector of number of pathogens spiked.
3. `mu`: overall mean
4. `theta1`, `theta2`: parameters of temperature and pH respectively
5. `taub1`, `taub2`: precision parameters for the distributions of the truncated polynomial coefficients for the spline functions
6. `tauL`: precision parameter for the laboratory random effects
7. `tauepsilon`: precision parameter for the ϵ_{ij}
8. `beta1[]`, `beta2[]`: parameters of the monomials of the spline functions
9. `b1[]`, `b2[]`: parameters of the truncated polynomials of the spline functions
10. `L[]`: vector of laboratory effects

References

[1] E.R. Berndt. *The Practice of Econometrics: Classical and and Contemporary.* Addison-Wesley, Reading, MA, 1991.

[2] S. Berry, R. J. Carroll, and D. Ruppert. Bayesian smoothing and regression splines for measurement error problems. *JASA*, 97:160–169, 2002.

[3] B. Brumback, D. Ruppert, and M. P. Wand. Comment on variable selection and function estimation in additive nonparametric regression using data-based prior by shively, kohn, and wood. *Journal of the American Statistical Association*, 94:794–797, 1999.

[4] C. M. Crainiceanu and D. Ruppert. Restricted likelihood ratio tests in nonparametric longitudinal models. *submitted*, 2004.

[5] C. M. Crainiceanu, D. Ruppert, J. R. Stedinger, and C. T. Behr. *Improving MCMC mixing for a GLMM Describing Pathogen Concentrations in Water Supplies. In Case Studies in Bayesian Statistics 6 (eds. C. Gatsonis, R.E. Kass, A. Carriquiry, A. Gelman, D. Higdon, D.K Pauler, I. Verdinelli).* Springer Verlag, New York, 2002.

[6] C. M. Crainiceanu, D. Ruppert, J. R. Stedinger, and C. T. Behr. Modeling the u.s. national distribution of waterborne pathogen concentrations with applications to *cryptosporidium parvum*. *Water Resources Research*, 39(9):1235–1249, 2003.

[7] P.H.C. Eilers and B.D. Marx. Flexible smoothing with b-splines and penalties. *Statistical Science*, 11(2):89–121, 1996.

[8] U.S. EPA. *National primary drinking water regulations: Long term enhanced surface water treatment rule.* Washington, D.C., 2001.

[9] R.L. Eubank. *Spline smoothing and Nonparametric Regression.* New York: Marcel Dekker, 1988.

[10] J. Fan and I. Gijbels. *Local Polynomial Modeling and its Applications.* Chapman and Hall, London, 1996.

[11] J.H. Friedman. Multivariate adaptive regression splines (with discussion). *The Annals of Statistics*, 19:1–141, 1991.

[12] A.E. Gelfand, S.K. Sahu, and B.P. Carlin. Efficient parameterizations for normal linear mixed model. *Biometrika*, 3:479–488, 1995.

[13] A.E. Gelfand, S.K. Sahu, and B.P. Carlin. *Efficient parameterizations for normal linear mixed model. In Bayesian Statistics 5 (eds. J.M. Bernardo, J. Berger, A.P. Dawid and A.F.M. Smith).* Oxford University Press, Oxford, 1995.

[14] P.J. Green and B.W. Silverman. *Nonparametric regression and Generalized Linear Models.* Chapman and Hall, London, 1994.

[15] J.E. Grizzle and D.M. Allan. Analysis of dose and dose response curves. *Biometrics*, 25:357–381, 1969.

[16] M.H. Hansen and C. Kooperberg. Spline adaptation in extended linear models (with discussion). *Statistical Science*, 17:2–51, 2002.

[17] T.J. Hastie and R. Tibshirani. *Generalized Additive Models.* Chapman and Hall, 1990.

[18] P. L. Meinhardt, D. P. Casemore, and K. B. Miller. Epidimiologic aspects of human cryptosporidiosis and the role of waterborne transmission. *Epidimil. Rev.*, 18(2):118–136, 1996.

[19] L. Ngo and M.P. Wand. Smoothing with mixed model software. *Journal of Statistical Software*, 9, 2004.

[20] R.T. Ogden. *Essential Wavelets for Statistical Applications and Data Analysis.* Birkhauser, Boston, 1996.

[21] D. Ruppert. Selecting the number of knots for penalized splines. *Journal of Computational and Graphical Statistics*, 2002.

[22] D. Ruppert, M.P. Wand, and R. Carroll. *Semiparametric Regression.* Cambridge University Press, 2003.

[23] J. Scheller, K. Connell, H. Shank-Givens, and C. Rodgers. *Design, implementation, and results of the EPA's 70-Utility ICR Laboratory Spiking Program in Information Collection Rule Data Analysis, edited by M. J. McGuire, J. McLain, and A. Obolensky.* AWWA Res. Found., Denver Colorado, 2002.

[24] D. Spiegelhalter, A. Thomas, and N. Best. WinBugs Version 1.3 User Manual, *Medical Research Council Biostatistics Unit, Cambridge*. http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/contents.shtml, 2000.

[25] M.E. Tarter and M.D. Lock. *Model-free Curve Estimation*. Chapman and Hall, New York, 1993.

[26] A. Ullah. Specification analysis of econometric models. *Journal of Quantitative Economics*, 2:187–209, 1985.

[27] G. Wahba. *Spline Models for Observational Data*. Wiley, 1990.

[28] M.P. Wand. Smoothing and mixed models. *submitted to Computational Statistics*, 2002.

[29] M.P. Wand and M.C. Jones. *Kernel Smoothing*. Chapman and Hall, London, 1995.

[30] Y. Wang. Mixed effects smoothing spline analysis of variance. *J. Royal Statistical Society, Series B*, 60:159–174, 1998.