Wood, S. N. (2016). Just Another Gibbs Additive Modeller: Interfacing JAGS and mgcv. Journal of Statistical Software, 75(7). DOI: 10.18637/jss.v075.i07

Journal of Statistical Software

December 2016, Volume 75, Issue 7. *doi: 10.18637/jss.v075.i07*

Just Another Gibbs Additive Modeler: Interfacing JAGS and mgcv

Simon N. Wood
University of Bristol

Abstract

The BUGS language offers a very flexible way of specifying complex statistical models for the purposes of Gibbs sampling, while its **JAGS** variant offers very convenient R integration via the **rjags** package. However, including smoothers in **JAGS** models can involve some quite tedious coding, especially for multivariate or adaptive smoothers. Further, if an additive smooth structure is required then some care is needed, in order to centre smooths appropriately, and to find appropriate starting values. R package **mgcv** implements a wide range of smoothers, all in a manner appropriate for inclusion in **JAGS** code, and automates centring and other smooth setup tasks. The purpose of this note is to describe an interface between **mgcv** and **JAGS**, based around an R function, `jagam`, which takes a generalized additive model (GAM) as specified in **mgcv** and automatically generates the **JAGS** model code and data required for inference about the model via Gibbs sampling. Although the auto-generated **JAGS** code can be run as is, the expectation is that the user would wish to modify it in order to add complex stochastic model components readily specified in **JAGS**. A simple interface is also provided for visualisation and further inference about the estimated smooth components using standard **mgcv** functionality. The methods described here will be un-necessarily inefficient if all that is required is fully Bayesian inference about a standard GAM, rather than the full flexibility of **JAGS**. In that case the **BayesX** package would be more efficient.

Keywords: R, BUGS, **JAGS**, additive model, spline, smooth, generalized additive mixed model.

1. Introduction

This paper is about automatically and reliably generating **JAGS** (Plummer 2003) model specification code and data implementing any generalized additive model (GAM, Hastie and Tibshirani 1990) that can be specified in the R (R Core Team 2016) package **mgcv** (Wood 2006, 2016). The purpose of this is to allow models with the complex smooth structure permitted

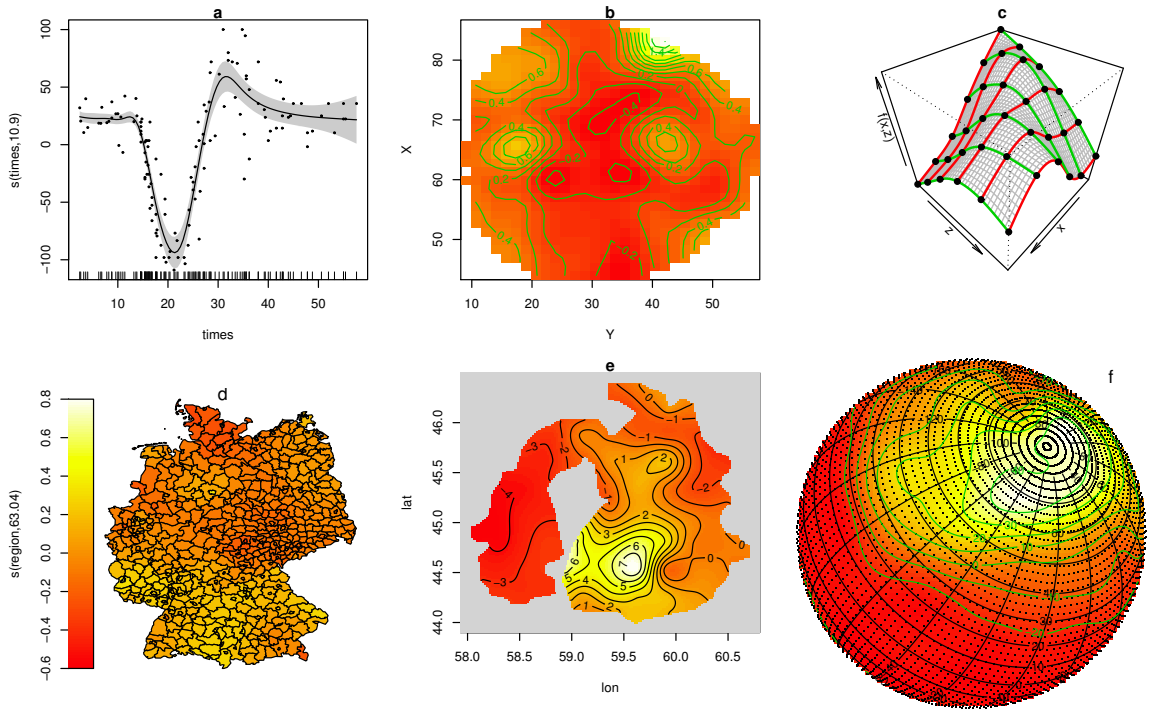

Figure 1: Some of the rich variety of smooths available in the **mgcv** package. From top right: A simple one-dimensional adaptive smooth; multidimensional thin-plate splines; multidimensional tensor product smooths; Gaussian Markov random fields; soap film finite area smoothers; splines on the sphere.

by **mgcv** (exemplified by Figure 1) combined with the complex random structure permitted by **JAGS** to be produced more easily than has hitherto been the case. As the paper's title makes clear, there is nothing new about using Markov chain Monte Carlo (MCMC) in general, or Gibbs sampling in particular, for smooth modeling. The paper's purpose is simply to make this easier and more automatic and hence less susceptible to implementation error, and to document the methods used to achieve this.

In principle, the **JAGS** package and language allows Bayesian inference about a very wide range of models that can be written as directed acyclic graphs (DAG). This class includes GAMs as one special case. The Bayesian view of spline smoothing and additive models is almost as old as splines and additive models themselves (Kimeldorf and Wahba 1970; Wahba 1983; Silverman 1985; Hastie and Tibshirani 2000; Fahrmeir and Lang 2001), and several authors have exploited this to use **JAGS** or BUGS (Spiegelhalter, Thomas, Best, and Gilks 1995) for generalized additive modeling, notably Crainiceanu, Ruppert, and Wand (2005) based on Ruppert, Wand, and Carroll (2003) and Zuur, Saveliev, and Ieno (2014).

In principle the **mgcv** package already included all the code required to set up smoothers for use with **JAGS**. This is because what is required is essentially the same as what is required to use any standard mixed modeling software for GAM inference: for example **mgcv** function `gamm` based on the appendix of Wood (2004) uses the **nlme** package (Pinheiro, Bates, DebRoy, Sarkar, and R Core Team 2016) in this way. However, a considerable degree of user expertise is required to implement this reliably in practice.

A particular area where difficulty can arise is in the use of centring constraints on model smooth components. Usually additive smooth model structures only make statistical sense if such constraints are applied (see e.g., Hastie and Tibshirani 1990), otherwise there is a global intercept associated with each smooth. However the **JAGS** requirement for all priors to be proper, means that failing to implement such constraints will not cause complete failure of Gibbs sampling. Instead one may see very wide credible intervals and poor mixing, but not realize that this is a model formulation problem rather than a statistical inevitability.

2. The `jagam` function

The new **mgcv** function `jagam` is designed to be called in the same way that the modeling function `gam` would be called. That is, a model formula and family object specify the required model structure, while the required data are supplied in a data frame or list or on the search path. However, unlike `gam`, `jagam` does no model fitting. Rather it writes **JAGS** code to specify the model as a Bayesian graphical model for simulation with **JAGS**, and produces a list containing the data objects referred to in the **JAGS** code, suitable for passing to **JAGS** via the **rjags** (Plummer 2016) function `jags.model`.

A simple model, with two univariate smooths and one tensor product smooth, exemplifies the approach. Suppose that we have a data frame, `dat`, containing the response and predictor variables, have loaded the **mgcv** package and have used `setwd` to set the working directory to something appropriate. The code

```
R> jd <- jagam(y ~ s(x0) + te(x1, x2) + s(x3), data = dat,
+    family = Gamma(link = log), file = "test.jags")
```

would specify a simple log gamma additive model structure,

$$\log(\mu_i) = f_1(x_{0i}) + f_2(x_{1i}, x_{2i}) + f_3(x_{3i}), \qquad y_i \sim \Gamma(\mu_i, \phi),$$

where f_2 is a scale invariant tensor product smoother, appropriate for representing smooth interaction terms. `jagam` returns a list containing standard **mgcv** GAM setup information (`pregam`) and a list, `jags.data`, containing the objects required by **JAGS** for model simulation. The function also writes a **JAGS** model specification in the file `test.jags`, as follows.

```
model {
  eta <- X %*% b
  for (i in 1:n) { mu[i] <-  exp(eta[i]) }
  for (i in 1:n) { y[i] ~ dgamma(r, r / mu[i]) }
  r ~ dgamma(.05, .005)
  scale <- 1 / r
  for (i in 1:1) { b[i] ~ dnorm(0, 0.00068) }
  K1 <- S1[1:9, 1:9] * lambda[1]  + S1[1:9, 10:18] * lambda[2]
  b[2:10] ~ dmnorm(zero[2:10], K1)
  K2 <- S2[1:24, 1:24] * lambda[3]  + S2[1:24, 25:48] * lambda[4] +
    S2[1:24, 49:72] * lambda[5]
  b[11:34] ~ dmnorm(zero[11:34], K2)
  K3 <- S3[1:9, 1:9] * lambda[6]  + S3[1:9, 10:18] * lambda[7]
```

```
  b[35:43] ~ dmnorm(zero[35:43], K3)
  for (i in 1:7) {
    lambda[i] ~ dgamma(.05, .005)
    rho[i] <- log(lambda[i])
  }
}
```

In fact code comments are also auto-generated, but have been removed from the above in accordance with journal requirements. They are designed to make it easy to locate the code relating to particular model components, and to draw attention to parts that the user might wish to modify.

In normal use the file would be edited to include the more complex stochastic components likely to have been the motivation for taking a Gibbs sampling approach. It can of course be used un-modified to simply simulate from the posterior of the model parameters, as in the following example code.

```
R> library("rjags")
R> jm <- jags.model("test.jags", data = jd$jags.data,
+    inits = jd$jags.ini, n.adapt = 2000, n.chains = 1)
R> sam <- jags.samples(jm, c("b", "rho", "scale"), n.iter = 10000, thin = 10)
```

The chains should then be checked for convergence and reasonable mixing in the standard ways. R package **coda** facilitates this (Plummer, Best, Cowles, and Vines 2006).

If all is in order, then many users would want to use the simulation output directly, but the utility function `sim2gam` can also be used to convert the simulation output into a reduced version of a fitted gam object, suitable for further use with standard **mgcv** functions. For example

```
R> jam <- sim2jam(sam, jd$pregam)
R> par(mfrow = c(1,3))
R> plot(jam)
```

yields Figure 2.

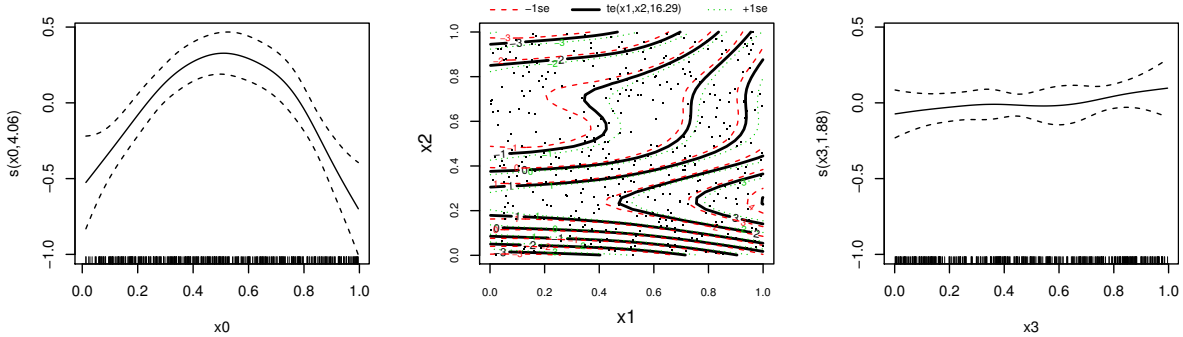

Figure 2: Plots of **JAGS** estimated smooth components of a log gamma additive model.

3. The underlying theory

3.1. Smoothers in JAGS

In **mgcv**, smooth functions are represented using spline like basis expansions, with quadratic penalties on the basis coefficients being used to avoid overfit. Generically a function $f(x)$ may be represented as

$$f(x) = \sum_{j=1}^{K} \beta_j b_j(x)$$

where the β_j are unknown model parameters/coefficients, and the $b_j(x)$ are spline like basis functions. K is chosen to be large enough to avoid oversmoothing, but small enough to avoid excessive computational cost. The fitted flexibility of f is controlled less by K than by the imposition, during fitting, of a quadratic smoothing penalty of the form

$$\sum_j \lambda_j \boldsymbol{\beta}^\top \mathbf{S}_j \boldsymbol{\beta},$$

where the $\mathbf{S}_j$ are matrices of known coefficients and the λ_j are smoothing parameters to be estimated. Often there is only a single component in the penalty, but the general summation form is necessary in order to implement adaptive and tensor product smooths, for example, and will also be used below to ensure the propriety of priors required for Gibbs sampling with **JAGS**. See Chapter 4 of Wood (2006) for more detail.

Kimeldorf and Wahba (1970), Wahba (1983) and Silverman (1985) provide the basis for viewing such smoothers in a Bayesian way, with the penalties induced by improper multivariate Gaussian priors, having precision matrices proportional to $\sum_j \lambda_j \mathbf{S}_j$. Adopting this viewpoint it is obvious that we can make inferences about smooths using Bayesian methods.

For practical Gibbs sampling in **JAGS** there are two cases to distinguish:

1. Those in which the prior precision matrix can be represented as a weighted sum of matrices that are all zero, apart from some unit entries on the leading diagonal, where no component matrices of the sum have unit entries in the same place.

2. Those in which the precision matrix can not be written in the above form.

Case 1 results in i.i.d. Gaussian priors on separate subsets of the parameters. For single smoothing parameter smooths it is possible to re-parameterize to achieve this case. Case 2 results in non-independent multivariate normal priors. In both cases the prior implied by the smoothing penalty is usually improper, as the penalty usually leaves some subspace of functions unpenalized. For Gibbs sampling with **JAGS** we require proper priors, but this is easily arranged.

Independent Gaussian prior smooths

Some smooths, such as the tensor product smoothers constructed by Wood, Scheipl, and Faraway (2013) or the truncated power basis P-splines advocated by Ruppert *et al.* (2003), automatically have penalties in which the $\mathbf{S}_j$ are identity matrices with some unit entries set to zero (and no unit entries in common between different $\mathbf{S}_j$). Let $\boldsymbol{\beta}_j$ denote the set

of coefficients for which the corresponding diagonal elements of $\mathbf{S}_j$ are 1 rather than zero. Then the prior on each element of $\boldsymbol{\beta}_j$ is $N(0, 1/\lambda_j)$ and the elements are independent a priori. A vague prior would typically be placed on λ_j. If $\boldsymbol{\beta}_0$ denotes the coefficients that are unpenalized, we can impose a prior $N(0, 1/\lambda_0)$ on these, where λ_0 is so small as to force the prior to be very vague, or λ_0 itself has a vague prior.

Any smooth that only has a single $\mathbf{S}_j$ and single λ_j can be re-parameterized to have a partial identity matrix prior precision matrix following Wood (2004). Dropping the index j, we find the symmetric eigen-decomposition $\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$. Suppose that the final M eigenvalues on the leading diagonal of $\boldsymbol{\Lambda}$ are zero (the remainder being positive). Define $\mathbf{D}$ to be the diagonal matrix with diagonal elements consisting of the square roots of the positive eigenvalues from $\boldsymbol{\Lambda}$, followed by M unit entries. If we now adopt the re-parameterization $\boldsymbol{\beta}' = \mathbf{D}\mathbf{U}^\top\boldsymbol{\beta}$ then the penalty matrix $\mathbf{S}$ becomes the identity matrix, with the last M leading diagonal elements set to zero. The corresponding model matrix is then $\mathbf{X}\mathbf{U}\mathbf{D}^{-1}$. The last M elements of $\boldsymbol{\beta}'$ are now un-penalized, and, as above, this can be handled by imposing independent $N(0, 1/\lambda_0)$ priors on these. The same result could be achieved somewhat more efficiently with a pivoted Cholesky decomposition. **mgcv** contains functions to perform this reparameterization automatically. Note that the preceding re-parameterization is slightly different to that employed in Crainiceanu *et al.* (2005), which starts from an indefinite $\mathbf{S}$, so that the reparameterization step also involves an element of approximation.

General Gaussian prior smooths

For a Gaussian likelihood, independent prior smooths can result in quite fast computation, because **JAGS** is then able to employ conjugate samplers. Similarly in generalized linear model settings, the samplers from the **JAGS** `glm` module can also lead to efficient computation. However outside these settings the independent prior approach is slow and block updates are preferable. In any case there are several important smoother classes that are not susceptible to writing in independent prior form, notably adaptive smooths and several types of tensor product smooth.

In fact implementing any quadratically penalized smoother in **JAGS** is straightforward, using `dmnorm`, the **JAGS** multivariate normal density. `dmnorm` is parameterized in terms of a precision matrix, for which $\sum_j \lambda_j \mathbf{S}_j$ can be used directly.

The only potential difficulty is that $\sum_j \lambda_j \mathbf{S}_j$ itself is usually rank deficient, implying an improper prior for the smooth. Again we must construct a prior for the null space of the smoothing penalty, but again it is possible to re-use existing **mgcv** facilities. Specifically, in the context of model selection, Marra and Wood (2011) propose a simple construction of a penalty on the null space related to the re-parameterization used in the previous section. Again use a symmetric eigen-decomposition $\sum_j \mathbf{S}_j = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$. Now let $\mathbf{U}_0$ denote the columns of $\mathbf{U}$ (eigenvectors) corresponding to zero eigenvalues. Let $\mathbf{S}_0 = \mathbf{U}_0\mathbf{U}_0^\top$. $\lambda_0\boldsymbol{\beta}^\top\mathbf{S}_0\boldsymbol{\beta}$ can be used to penalize the null space of the smoother by adding $\lambda_0\mathbf{S}_0$ to the precision matrix, hence making the prior on $\boldsymbol{\beta}$ proper. **mgcv** can generate such null space penalties automatically.

Smoothing parameter priors

`jagam` automates two possibilities for smoothing parameter priors: vague gamma priors on the λ_j, or bounded uniform priors on $\rho_j = \log \lambda_j$. The former will be conjugate in a fully Gaussian setting, but the latter may be considered more interpretable for the user used to

thinking about log smoothing parameters.

3.2. Centring the smoothers

As constructed so far, each smooth in an additive model would include its own global intercept. The data provides no information to identify these multiple intercepts, so they are only formally identifiable because of the priors put on them, which are vague priors of convenience. This lack of statistically meaningful identifiability will serve to substantially inflate credible intervals and promote slow mixing, so it is preferable to remove the redundant intercepts from the model. In an additive model context this is usually done by centring the smooths (Hastie and Tibshirani 1986, 1990; Chambers and Hastie 1991; Wood 2006). That is we impose the condition that each smooth should sum to zero over the observed values its covariates. i.e., $\sum_{i=1}^{n} f(x_i) = 0$. Other constraints are possible, but generally give wider credible intervals for the constrained smooths (see Section 4 of Wood *et al.* 2013, for a discussion). **mgcv** has facilities to simply absorb centring constraints into the basis by reparameterization, as described in section 4.2 of Wood (2006). This absorption is done before any reparameterization or construction of priors on the null space.

3.3. Initial values

In the Gaussian likelihood case, with gamma priors for the smoothing parameters, the initial values of the model coefficients and smoothing parameters are rather unimportant. In this situation conjugate samplers are used and, although poor starting values may prolong burn-in, eventually good results will be obtained.

Beyond the simple Gaussian context, more care is needed, since poor starting values can lead to poor tuning of non-conjugate samplers, and a consequent failure to properly explore the region of high posterior probability (along then with high sensitivity to the parameters of the smoothing parameter priors). `jagam` adopts the **mgcv** default smoothing parameter initializations and then performs one step of the penalized iteratively re-weighted least squares method for GAM fitting, in order to obtain starting values for the coefficients which are compatible with the initial smoothing parameters. The initial coefficients and corresponding standard errors are also used to set the scale of any required uninformative priors on the model coefficients: the prior standard deviation is set to 10 times the sum of the absolute value of the initial coefficient estimate and its standard error.

3.4. Further inference

Having setup a GAM for use in **JAGS** and simulated from it, the user will typically want to visualize the smooths and predict from them. In addition some notion of the effective degrees of freedom of the smooth is useful.

An obvious way to visualize the smooths is to draw curves from the posterior, and either compute appropriate pointwise quantiles in order to produce credible intervals, or to simply plot the curves. Examples are given below. Alternatively, smooths may be plotted with 'two standard error bands', in the manner introduced in Hastie and Tibshirani (1990). To this end it is only necessary to compute the mean coefficients from the simulation, to use in place of coefficient estimates, $\hat{\boldsymbol{\beta}}$, and to compute the observed covariance matrix of the simulated coefficients, $\mathbf{V}_\beta$, from which the standard error bands are readily computed. In fact $\hat{\boldsymbol{\beta}}$ and

$\mathbf{V}_\beta$ also complete the preliminary `gam` object produced by `jagam` sufficiently for prediction using `predict.gam`.

Finally some notion of the effective degrees of freedom of the model and its component smooths is useful. In a simple Gaussian additive model context a measure of the model effective degrees of freedom is $\mathrm{tr}(\mathbf{F})$ where $\mathbf{F} = (\mathbf{X}^\top\mathbf{X} + \sum_j \lambda_j \mathbf{S}_j)^{-1}\mathbf{X}^\top\mathbf{X}$. In the generalized additive model context $\mathbf{F} = (\mathbf{X}^\top\mathbf{W}\mathbf{X} + \sum_j \lambda_j \mathbf{S}_j)^{-1}\mathbf{X}^\top\mathbf{W}\mathbf{X}$, where $\mathbf{W}$ is the diagonal matrix of iteratively re-weighted least squares weights used in fitting. In the presence of random effects it is better to use $\mathbf{F} = \mathbf{V}_\beta\mathbf{X}^\top\mathbf{W}\mathbf{X}/\phi$, in which $\mathbf{W}$ is the diagonal IRLS weight matrix with the random effects set to their posterior expectations and ϕ is the scale parameter or its estimate. When $\mathbf{V}_\beta$ is computed by simulation then this latter definition has the advantage of including a component for smoothing parameter uncertainty, however to compute it in practice requires that the expected value of the response, `mu`, be monitored during simulation. See chapter 4 of Wood (2006) for further discussion.

Given $\mathbf{F}$, then the effective degrees of freedom of component smooths are obtained by summing the leading diagonal elements of $\mathbf{F}$ corresponding to the coefficients of the smooth concerned. Notice that under substantial modification of the `jagam` template model (involving modification of the response distribution, for example), the weighted versions of $\mathbf{F}$ may make no sense. It may then be better to fall back on the effective degrees of freedom that would have been computed if the model were a simple Gaussian additive model, or to use the estimate proposed by Plummer (2002).

4. Examples

As two simple examples consider the union wages example and the Sitka growth example from Crainiceanu *et al.* (2005). Both datasets are available in the **SemiPar** R package (Wand 2014). Loading the **JAGS** `glm` module, via `load.module("glm")` improves the efficiency of both examples in this section.

4.1. The union wages data

The data frame `trade.union` contains a binary indicator of whether or not a worker is a trade union member, along with their hourly wage in US dollars. Consider the simple logistic regression model

$$\mathrm{logit}(p_i) = f(\texttt{wage}_i), \qquad \texttt{union.member}_i \sim \mathrm{Bernoulli}(p_i)$$

where smooth function f is represented by a rank 20 thin plate regression spline. A `jagam` call sets the model up

```
R> jd <- jagam(union.member ~ s(wage, k = 20), data = trade.union,
+    family = binomial, file = "union.jags")
```

resulting in the following **JAGS** model specification file (again the auto-generated comments have been removed to comply with journal requirements).

```
model {
  eta <- X %*% b
```

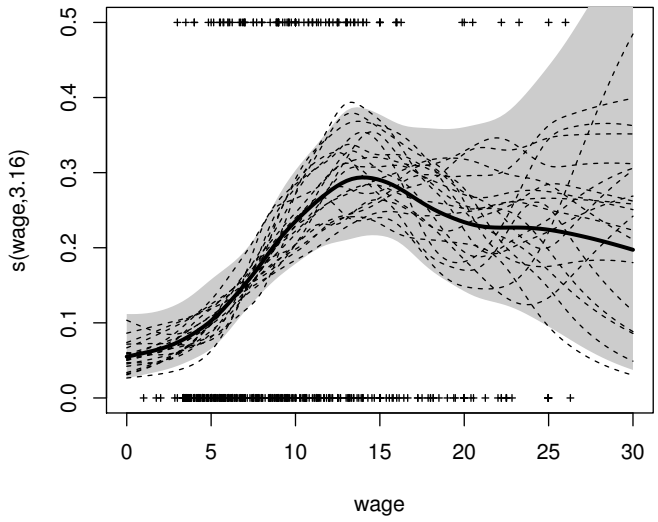

Figure 3: Results for the Union Wages model.

```
  for (i in 1:n) { mu[i] <-  ilogit(eta[i]) }
  for (i in 1:n) { y[i] ~ dbin(mu[i], w[i]) }
  for (i in 1:1) { b[i] ~ dnorm(0, 0.018) }
  K1 <- S1[1:19, 1:19] * lambda[1]  + S1[1:19, 20:38] * lambda[2]
  b[2:20] ~ dmnorm(zero[2:20], K1)
  for (i in 1:2) {
    lambda[i] ~ dgamma(.05, .005)
    rho[i] <- log(lambda[i])
  }
}
```

The following commands then compile and simulate from the model.

```
R> library("rjags")
R> load.module("glm")
R> jm <- jags.model("union.jags", data = jd$jags.data,
+    inits = jd$jags.ini, n.chains = 1)
R> sam <- jags.samples(jm, c("b", "rho", "mu"), n.iter = 10000, thin = 10)
```

On a 3GHz mid range laptop computer, simulation took 18 seconds, yielding effective sample sizes averaging around 400 for `rho` and 800 for `b` and `mu`, for the 1000 samples stored. Crainiceanu *et al.* (2005) report around 9 minutes for this model (albeit with a slightly different smoothing penalty) for the same simulation length, on a 3.6GHz PC, although they do not report effective sample sizes, so the comparison is not completely straightforward. Note that failing to supply starting values greatly increases the adaptation and burn in time required to achieve reliable results for this example.

Finally a partial `gam` object can be created for convenient plotting and prediction. The following then produces a plot of the modeled probability of union membership against wages with a credible interval, along with a visualization of the union membership data. The interval is wide at high wages, failing to provide a very useful indication of the range of smooth shapes

compatible with the data, so the following code also adds a sample of 20 curves from the posterior.

```
R> jam <- sim2jam(sam, jd$pregam)
R> plot(jam, shade = TRUE, shift = coef(jam)[1], trans = binomial()$linkinv,
+    rug = FALSE, ylim = c(-100, -coef(jam)[1]), seWithMean = TRUE,
+    xlim = c(0, 30), lwd = 3)
```

Given the basic plot, now add the original union membership data.

```
R> nu <- trade.union$union.member == 0
R> with(trade.union, points(wage[nu], 0 * wage[nu], pch = 3, cex = .5))
R> with(trade.union, points(wage[!nu], 0 * wage[!nu] + .5, pch = 3, cex = .5))
```

And now add 20 smooth curves drawn from the posterior, to examine variability in the smooth shape.

```
R> ii <- 1:20 * 50
R> pd <- data.frame(wage = 0:300 / 10)
R> Xp <- predict(jam, type = "lpmatrix", newdata = pd)
R> for (i in ii) {
+    p <- binomial()$linkinv(Xp %*% sam$b[, i, 1])
+    lines(pd$wage, p, lty = 2)
+  }
```

The result is shown in Figure 3, indicating that the peak in the probability curve is not a very robust feature, and it would be difficult to rule out a monotonic relationship between wages and probability of union membership.

4.2. The Sitka growth data

This example illustrates the modification of an auto-generated **JAGS** model file to implement random effects. The 'sitka' data contain repeated measurements over time of log size for Sitka spruce saplings grown under conditions of enhanced ozone, or control conditions. A simple model has a smooth effect for time, a random intercept for each tree and an ozone effect,

$$\log(\texttt{size}_i) = \alpha + f(\texttt{days}_i) + \beta\texttt{ozone}_i + b_{j(i)} + \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$, $b_j \sim N(0, \sigma_b^2)$, and $j(i)$ is the index of the tree from which the i^{th} measurement is taken. Everything is Gaussian, so a fully conjugate setup can be employed, and it is worth diagonalizing the smoothing penalties.

```
R> jd <- jagam(log.size ~ s(days) + ozone, data = sitka,
+    file = "sitka0.jags", diagonalize = TRUE)
```

creates a default **JAGS** model file which can be modified to include the random effect as follows, where the gray italic code has been added to the non-italic auto-generated **JAGS** code (and again to meet journal requirements the auto-generated comments have been removed).

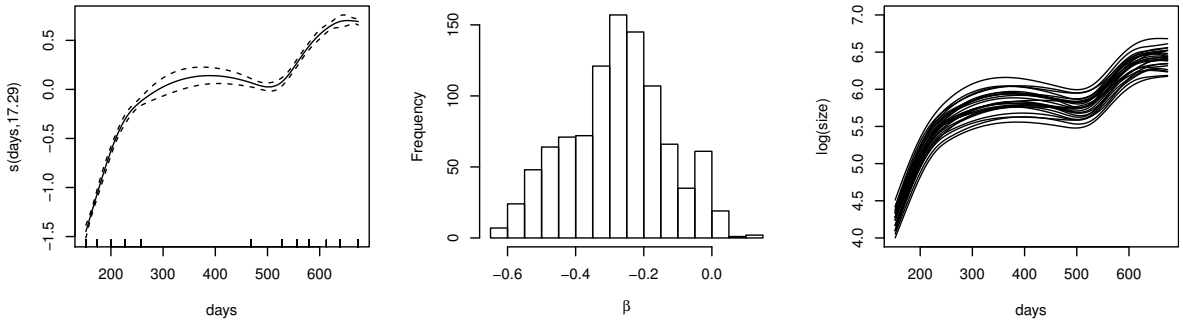

Figure 4: Results for the Sitka growth model.

```
model {
  mu0 <- X %*% b
  for (i in 1:n) { mu[i] <- mu0[i] + d[id[i]] }
  for (i in 1:n) { y[i] ~ dnorm(mu[i], tau) }
  scale <- 1 / tau
  tau ~ dgamma(.05, .005)
  for (i in 1:nd) { d[i] ~ dnorm(0, taud) }
  taud ~ dgamma(.05, .005)
  for (i in 1:2) { b[i] ~ dnorm(0, 3e-04) }
  for (i in 3:10) { b[i] ~ dnorm(0, lambda[1]) }
  for (i in 11:11) { b[i] ~ dnorm(0, lambda[2]) }
  for (i in 1:2) {
    lambda[i] ~ dgamma(.05, .005)
    rho[i] <- log(lambda[i])
  }
}
```

The modification involves adding a tree specific random effect, `d`, to the linear predictor. Notice the need to add `id`, the vector attributing measurements to trees, and `nd`, the number of trees, to the **JAGS** data. The following code compiles and simulates from the model, produces a default plot of the smooth effect of time (given the sum to zero identifiability constraint), a histogram of draws from the posterior for β and illustration of 25 curves, $\alpha + f(\texttt{days})$, drawn from the posterior.

```
R> jd$jags.data$id <- sitka$id.num
R> jd$jags.data$nd <- length(unique(sitka$id.num))
R> jm <- jags.model("sitka.jags", data = jd$jags.data,
+    inits = jd$jags.ini, n.chains = 1)
R> sam <- jags.samples(jm, c("b", "rho", "scale", "mu"),
+    n.iter = 10000, thin = 10)
R> jam <- sim2jam(sam, jd$pregam)
R> plot(jam, pages = 1)
R> hist(sam$b[2, , 1])
R> days <- 152:674
```

```
R> pd <- data.frame(days = days, ozone = days * 0)
R> Xp <- predict(jam, newdata = pd, type = "lpmatrix")
R> ii <- 1:25 * 20 + 500
R> for (i in 1:25) {
+    fv <- Xp %*% sam$b[, ii[i], 1]
+    if (i == 1) {
+      plot(days, fv, type = "l", ylim = c(4, 7))
+    } else {
+      lines(days, fv)
+    }
+  }
```

The results are shown in Figure 4. Notice how the left hand plot, which shows the credible interval for f *subject to constraint*, suggests a very limited range of shapes for f. This is born out by the right hand plot, in which most of the variability in the curves is in their level, rather than their shape.

5. Conclusion

The **JAGS** software offers enormous flexibility in the specification of complex random effects structures. Incorporating spline type smoothers into such models is routine, but somewhat tedious to code on a case by case basis, as well as being prone to error, especially for smooths of several variables. The `jagam` function offers a useful automation of the process of incorporating any smooth built into **mgcv** into a **JAGS** model, while dealing seamlessly with initialization and centring constraints and allowing straightforward posterior prediction.

The main disadvantage of the approach is computational speed. Gibbs sampling for these models can be slow, especially if covariates are correlated. Indeed if only simple random effects are required then the random effects already available in **mgcv** may be much faster computationally. Similarly **BayesX** (see e.g., Fahrmeir and Lang 2001; Fahrmeir, Kneib, and Lang 2004; Umlauf, Adler, Kneib, Lang, and Zeileis 2015) is a substantially more efficient route to fully Bayesian inference with GAMs if the flexibility of **JAGS** is not required, while Stan (Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, and Riddell 2017) offers another alternative likely to offer efficiency advantages. In these correlated settings it is also likely that Hamiltonian Monte Carlo methods (e.g., Girolami and Calderhead 2011) would enhance efficiency.

Acknowledgments

I am grateful to Ciprian Crainiceanu and Mirjam Barrueto for some very helpful discussion of Crainiceanu *et al.* (2005), and to two referees for pointing out how much the `glm` library can improve performance, suggestions on the 'fixed effect' priors and other useful comments. This work was funded by UK EPSRC grant EP/K005251/1.

References

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

Chambers JM, Hastie TJ (1991). *Statistical Models in S*. Chapman & Hall/CRC.

Crainiceanu CM, Ruppert D, Wand MP (2005). "Bayesian Analysis for Penalized Spline Regression Using **WinBUGS**." *Journal of Statistical Software*, **14**(14), 1–24. doi:10.18637/jss.v014.i14.

Fahrmeir L, Kneib T, Lang S (2004). "Penalized Structured Additive Regression for Space-Time Data: A Bayesian Perspective." *Statistica Sinica*, **14**(3), 731–761. doi:10.1002/bimj.200490310.

Fahrmeir L, Lang S (2001). "Bayesian Inference for Generalized Additive Mixed Models Based on Markov Random Field Priors." *Journal of the Royal Statistical Society C*, **50**(2), 201–220. doi:10.1111/1467-9876.00229.

Girolami M, Calderhead B (2011). "Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods." *Journal of the Royal Statistical Society B*, **73**(2), 123–214. doi:10.1111/j.1467-9868.2010.00765.x.

Hastie T, Tibshirani R (1986). "Generalized Additive Models." *Statistical Science*, **1**(3), 297–318. doi:10.1214/ss/1177013609.

Hastie T, Tibshirani R (1990). *Generalized Additive Models*. Chapman & Hall.

Hastie T, Tibshirani R (2000). "Bayesian Backfitting." *Statistical Science*, **15**(3), 196–223. doi:10.1214/ss/1009212815.

Kimeldorf GS, Wahba G (1970). "A Correspondence between Bayesian Estimation on Stochastic Processes and Smoothing by Splines." *The Annals of Mathematical Statistics*, **41**(2), 495–502. doi:10.1214/aoms/1177697089.

Marra G, Wood SN (2011). "Practical Variable Selection for Generalized Additive Models." *Computational Statistics & Data Analysis*, **55**(7), 2372–2387. doi:10.1016/j.csda.2011.02.004.

Pinheiro J, Bates D, DebRoy S, Sarkar D, R Core Team (2016). ***nlme**: Linear and Nonlinear Mixed Effects Models*. R package version 3.1-128, URL https://CRAN.R-project.org/package=nlme.

Plummer M (2002). "Discussion of the Paper by Spiegelhalter et al." *Journal of the Royal Statistical Society B*, **64**(4), 620–621.

Plummer M (2003). "**JAGS**: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In K Hornik, F Leisch, A Zeileis (eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Vienna, Austria. ISSN 1609-395X. URL http://www.ci.tuwien.ac.at/Conferences/DSC-2003/.

Plummer M (2016). ***rjags**: Bayesian Graphical Models Using MCMC*. R package version 4-6, URL https://CRAN.R-project.org/package=rjags.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://CRAN.R-project.org/doc/Rnews/.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

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

Silverman BW (1985). "Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting." *Journal of the Royal Statistical Society B*, **47**(1), 1–53.

Spiegelhalter DJ, Thomas A, Best NG, Gilks WR (1995). "**BUGS**: Bayesian Inference Using Gibbs Sampling. Version 0.50." *MRC Biostatistics Unit, Cambridge*.

Umlauf N, Adler D, Kneib T, Lang S, Zeileis A (2015). "Structured Additive Regression Models: An R Interface to **BayesX**." *Journal of Statistical Software*, **63**(21), 1–46. doi:10.18637/jss.v063.i21.

Wahba G (1983). "Bayesian Confidence Intervals for the Cross-Validated Smoothing Spline." *Journal of the Royal Statistical Society B*, **45**(1), 133–150.

Wand M (2014). ***SemiPar**: Semiparametic Regression*. R package version 1.0-4.1, URL https://CRAN.R-project.org/package=SemiPar.

Wood SN (2004). "Stable and Efficient Multiple Smoothing Parameter Estimation for Generalized Additive Models." *Journal of the American Statistical Association*, **99**(467), 673–686. doi:10.1198/016214504000000980.

Wood SN (2006). *Generalized Additive Models: An Introduction with R*. Chapman & Hall/CRC, Boca Raton.

Wood SN (2016). ***mgcv**: Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation*. R package version 1.8-16, URL https://CRAN.R-project.org/package=mgcv.

Wood SN, Scheipl F, Faraway JJ (2013). "Straightforward Intermediate Rank Tensor Product Smoothing in Mixed Models." *Statistics and Computing*, **23**(3), 341–360. doi:10.1007/s11222-012-9314-z.

Zuur AF, Saveliev AA, Ieno EN (2014). *A Beginner's Guide to Generalized Additive Mixed Models with R*. Highland Statistics.

Affiliation:

Simon N. Wood
School of Mathematics
University of Bristol
Bristol BS8 1TW, United Kingdom
E-mail: Simon.Wood@R-project.org

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
December 2016, Volume 75, Issue 7 *Submitted:* 2014-11-30
doi:10.18637/jss.v075.i07 *Accepted:* 2015-10-12