An Implementation of Bayesian Adaptive Regression Splines (BARS) in C with S and R Wrappers

Garrick Wallstrom
Department for Biomedical Informatics
University of Pittsburgh
Pittsburgh, PA, 15213, USA

Jeffrey Liebner and Robert E. Kass
Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213, USA

SUMMARY

BARS (DiMatteo, Genovese, and Kass, 2001, *Biometrika*) uses the powerful reversible-jump MCMC engine to perform spline-based generalized nonparametric regression. It has been shown to work well in terms of having small mean-squared error in many examples (smaller than known competitors), as well as producing visually-appealing fits that are smooth (filtering out high-frequency noise) while adapting to sudden changes (retaining high-frequency signal). However, BARS is computationally intensive. The original implementation in S was too slow to be practical in certain situations, and was found to handle some data sets incorrectly. We have implemented BARS in C for the Normal and Poisson cases, the latter being important in neurophysiological and other point-process applications. The C implementation includes all needed subroutines for fitting Poisson regression, manipulating B-splines (using code created by Bates and Venables), and finding starting values for Poisson regression (using code for density estimation created by Kooperberg). The code utilizes only freely-available external libraries (LAPACK and BLAS) and is otherwise self-contained. We have also provided wrappers so that BARS can be used easily within S or R.

Key words: Curve-Fitting; Free-Knot Splines; Nonparametric Regression; Peri-Stimulus Time Histogram (PSTH); Poisson Process.

1 Introduction

The problem of curve-fitting is one of estimating a function $y = f(x)$ from data pairs (x_j, y_j), and it is usually formalized by taking the response values y_j to be observations from random variables Y_j assumed to arise from the "signal plus noise" model

$$Y_j = f(x_j) + \varepsilon_j. \tag{1}$$

When $f(x)$ is allowed to have a flexible form, this becomes the problem of nonparametric regression, and it is usually solved by some variation on least squares, which may be motivated by an assumption that the ε_j variables are independent and Normally distributed. In many applications it may be preferable to replace the assumption of additive, Normal noise in (1) with a more general stochastic response specification. The example that concerns us most is Poisson nonparametric regression, and in our applications the explanatory variable x becomes time t and we write

$$\begin{aligned} Y_j &\sim P(y_j|\lambda_j) \\ \log \lambda_j &= f(t_j). \end{aligned} \tag{2}$$

The purpose of this paper is to describe software for fitting models of the forms (1) or (2) using a method called Bayesian Adaptive Regression Splines (BARS; DiMatteo, Genovese, and Kass, 2001).

Our interest in this problem comes from extensive experience in fitting curves to neurophysiological data (Kass, Ventura, and Brown, 2005; Kass, Ventura, and Cai, 2003; Ventura *et al.*, 2002). A typical data display is given in Figure 1. The raw data are neuronal firing event times in response to many repititions of the same stimulus, and the firing events are usually aggregated across the repititions or "trials" to produce event counts within small time bins. These are then typically normalized and displayed as a Peri-Stimulus Time Histogram (PSTH). The PSTH is supposed to capture the neuronal response to the stimulus and our goal is to smooth it. In some of our work we used kernel smoothers (Ventura *et al.*, 2002), but we came across neurons like the one displayed in Figure 1 where kernel smoothers are unable to capture the shape of the firing response. The typical situation is that the stimulus-response of the neuron may reasonably be assumed to be slowly varying throughout most of the time interval of observation, but may be subject to a sudden onset of activity. We wanted to use a statistical method that would capture the onset of activity but would allow the response in the remainder of the time interval to be smooth. The BARS fit shown in Figure 1 has the desired behavior.

The essential idea behind BARS is to assume in (1) that $f(x)$ is a cubic spline, and to determine the number of knots and their locations by applying reversible-jump MCMC. A substantial literature has documented the effectiveness of spline-based nonparametric regression and its generalizations, with the knots being determined empirically. See

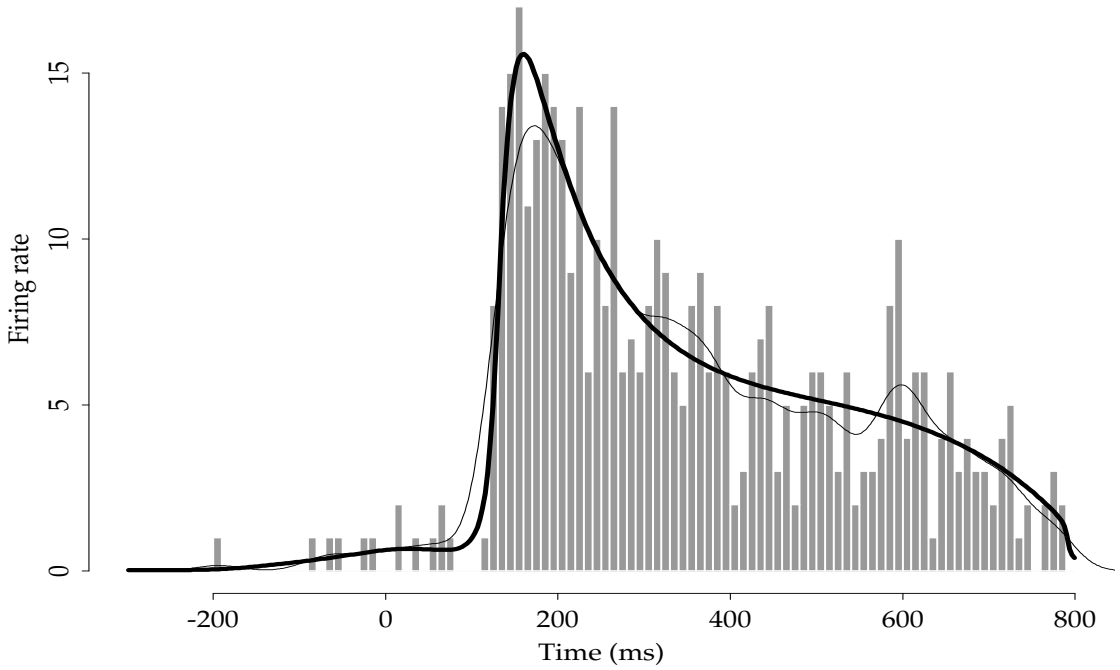

Figure 1: *Normalized histogram and fits using a kernel density estimator (thin line) and BARS (thick line). Units are in spiking events per second, the usual units for neuronal firing-rate functions based on neuronal spiking events. The data come from a neuron in inferotemporal cortex recorded during 16 replications (in physiological jargon, 16 trials) of an experiment described by Baker* et al.*, 2002. Time 0 corresponds to initiation of a visual stimulus in the neuron's receptive field during a passive fixation task. The neuronal firing rate is assumed to be smooth. The kernel density estimate, with bandwidth selected according to Equation (5.5) of Venables and Ripley (1999), misses the sudden jump at about 125 milliseconds and is generally too variable. BARS succeeds in capturing the early jump, and is otherwise suitably smooth. (It is possible that there is a tendency for the neuron to oscillate, with a small peak apparent at around 600 milliseconds. With a prior that puts weight on large numbers of knots BARS can fit this small peak, and it does so fairly reliably in test simulations. However, from a neurophysiological perspective, it is reasonable to assume that when the firing-rate is averaged across trials it varies slowly over most of its domain; a careful consideration of this phenomenon would require a within-trial analysis.)*

Hansen and Kooperberg (2002), and the accompanying discussion. The idea of applying reversible-jump MCMC to this problem is due to Dennison, Mallick, and Smith (1998) but there were serious defects in their implementation (DiMatteo, *et al.*, 2001; Kass and Wallstrom, 2002). DiMatteo *et al.* showed in several examples that BARS could perform dramatically better than the Dennison *et al.* version, and also another closely-related spline approach, which themselves were better in many examples than competitors such as wavelet-based methods. BARS has been used in variety of applications in neurophysiology, imaging, genetics, and EEG analysis. (See Behseta and Kass, 2005; Behseta, Kass, and Wallstrom, 2005; Behseta, Kass, Moorman, and Olson, 2007; DiMatteo *et al.*, 2001; Kass, Ventura, and Cai, 2003; Kass and Wallstrom, 2002; Zhang *et al.*, 2003; Wallstrom *et al.*, 2004.) Furthermore, preliminary results have indicated that, with reasonable sample sizes, posterior credible regions produced by BARS have very close to the correct frequentist coverage probabilities. For example, simulating from a curve based on the firing-rate function displayed in Figure 1, the coverage of the 95% posterior credible region for location of the peak was 95.9%, with two-digit accuracy (simulation $SE = .005$, based on 7,600 simulated data sets). A referee has also mentioned that, as simulation-based procedure, BARS may be less sensitive than deterministic methods to very small perturbations of the data—as produced by moving data across computing platforms. We have checked our results using BARS across multiple computing platforms and have obtained highly reproducible results.

The original implementation of DiMatteo *et al.* was in S. It was very slow, and also suffered from bugs that caused occasional very poor fits. We wished to improve the implementation. In addition, in order to study frequentist coverage probabilities we had to use a parallel array of multiple processors, and therefore needed self-contained code. This article discusses a relatively fast, self-contained, and, we believe, careful BARS implementation in C with S and R wrappers. The R code has been contributed as a CRAN package. We give an overview of BARS in Section 2 and an overview of the code in Section 3. Section 4 describes the S and R wrappers and Section 5 gives pseudo-code for the C implementation.

An important point about our implementation concerns our application to smoothing a PSTH. While the PSTH is called a "histogram," it may not be obvious that it is in any sense estimating a density function. The fact that the PSTH may be considered a density estimator is related to a fundamental property of Poisson processes, as we will review briefly below. In terms of implementation, this allows us to use the LOGSPLINE code for density esimation developed by Kooperberg to provide starting values in our Poisson nonparametric regression code. We have found that the more computationally-intensive BARS consistently improves on LOGSPLINE, and it has become a routine method for us in analyzing neuronal data.

2 Overview of BARS

Consider the problem of making inferences about a function $f(t)$, where t lies in an interval $[A, B]$, based on data $y = y_1, \ldots, y_n$ obtained at $t = t_1, \ldots, t_n$, with each Y_j assumed to depend probabilistically on $f(t_j)$ (and, following the usual convention, y_j represents the observed value of the random variable Y_j). To solve this problem BARS fits the spline-based generalized nonparametric regression model for data Y_j depending on a variable t, We write $Y_j \sim p(y_j|\theta_j, \zeta)$ to indicate that Y_j follows some family of distributions with density $p(y_j|\theta_j, \zeta)$ and we write the generalized nonparametric regression problem in the form

$$\begin{aligned} Y_j &\sim p(y_j|\theta_j, \zeta) \\ \theta_j &= f(x_j). \end{aligned} \tag{3}$$

with f being a spline having knots at unknown locations $\xi_1, \ldots, \xi_k$. Model (3) includes a vector of nuisance parameters ζ to indicate generality, though in the Poisson case there are no nuisance parameters. Writing $f(t)$ in terms of basis functions $b_{\xi,h}(t)$ as $f(t) = \sum_h b_{\xi,h}(t)\beta_{\xi,h}$ the function evaluations $f(t_1), \ldots, f(t_n)$ may be collected into a vector $(f(t_1), \ldots, f(t_n))^T = X_\xi \beta_\xi$, where X_ξ is the design matrix and β_ξ is the coefficient vector. For a given knot set $\xi = (\xi_1, \ldots, \xi_k)$ model (3) poses a relatively easy estimation problem; for exponential-family responses (such as Poisson) it becomes a generalized linear model. Selecting the interval $[A, B]$ can be difficult due to spline boundary conditions (see Hansen and Kooperberg, 2002, and the accompanying discussion), but in our own work we have found simple heuristics (such as extending beyond the data by a fixed time interval) to be satisfactory; in any case, our software assumes a choice of interval $[A, B]$ has been made. The remaining hard part of the problem is determining the knot set ξ, and using the data to do so provides the ability to fit a wide range of functions (as reviewed by Hansen and Kooperberg, 2002). BARS is an MCMC-based algorithm that samples from a suitable approximate posterior distribution on the knot set ξ. This, in turn, produces samples from the posterior on the space of splines. In practice, cubic splines and the natural spline basis have been used in most applications. BARS could be viewed as a powerful engine for searching for an "optimal" knot set, but because it generates a posterior on the space of splines it produces an improved spline estimate based on model averaging (e.g., Kass and Raftery, 1995) and it also provides uncertainty assessments.

2.1 MCMC in BARS

Key features of the MCMC implementation of BARS include (i) a reversible-jump chain on ξ after integrating the marginal density

$$p(y|\xi) = \int p(y|\beta_\xi, \xi, \zeta)\pi(\beta_\xi, \zeta|\xi)d\beta_\xi d\zeta \tag{4}$$

(where $y = (y_1, \ldots, y_n)$), the integration being performed exactly for Normal data and approximately, by Laplace's method, otherwise, (ii) continuous proposals for ξ, and (iii) a locality heuristic for the proposals that attempts to place potential new knots near existing knots. For notational convenience here we are supressing the dependence of the knot set ξ on the number of knots k but BARS explores the space of generalized regression models defined by ξ and k and the prior on k can, in some cases, control the algorithm in important ways (see DiMatteo, *et al.*, 2001, Hansen and Kooperberg, 2002, and Kass and Wallstrom, 2002).

The use of reversible-jump MCMC to select knots was suggested by Denison *et al.* (1998), following the lead of Green (1995), who discussed the special case of change-point problems. However, aspects of BARS outlined in (i)-(iii) distinguish it from (and improve upon) the method of Denison *et al.* (see Kass and Wallstrom, 2002). The first implementation feature, item (i) above, introduces an analytical step within the MCMC partly to simplify the problem of satisfying detailed balance and partly for the sake of MCMC efficiency (which is generally increased when parameters are integrated; see Liu, Wong, and Kong, 1994). In addition, BARS takes advantage of the high accuracy of Laplace's method in this context. In doing so the "unit-information" prior discussed by Kass and Wasserman (1995) and Pauler (1998) has been used (as π in (4)), and this gives the interpretation that the algorithm is essentially using BIC to define a Markov chain on the knot sets. The importance of performing the integral (4), at least approximately, has been stressed by Kass and Wallstrom (2002). Continuous proposals and the locality heuristic (items (ii) and (iii)) together allow knots to be placed close to one another, which is advantageous when there is a sudden jump in the function.

For each draw $\xi^{(g)}$ from the posterior distribution of ξ, a draw $\beta_\xi^{(g)}$ is obtained from the conditional posterior of β_ξ, conditionally on $\xi^{(g)}$. The conditional posterior of β_ξ often may be assumed Normal, but in some cases the Normal approximation is not very good. DiMatteo, *et al.* described an importance reweighting scheme to improve upon the Normal approximation. In the code discussed here, if the Normal approximation seems poor, we instead use a conditional Metropolis update.

From $\beta_\xi^{(g)}$ we obtain fitted values $f^{(g)}(\tilde{t}) = \sum b_{\xi,h}(\tilde{t})\beta_{\xi,h}^{(g)}$ for selected $\tilde{t}$ and these, in turn, may be used to produce a draw $\phi^{(g)}$ from the posterior distribution of any characteristic $\phi = \phi(f)$ (such as the value at which the maximum of $f(t)$ occurs). Thus, the key output of BARS is the set of vectors $\tilde{f}^{(g)} = (f^{(g)}(\tilde{t}_1), f^{(g)}(\tilde{t}_2), \ldots, f^{(g)}(\tilde{t}_p))$ for MCMC iterates $g = 1, \ldots, G$, each $\tilde{f}^{(g)}$ being a vector of fits along a grid $\tilde{t}_1, \tilde{t}_2, \ldots, \tilde{t}_p$ that suitably covers the interval $[A, B]$. The user may sample from the posterior distribution of any functional ϕ simply by evaluating $\phi^{(g)} = \phi(\tilde{f}^{(g)})$. For instance, a sample from the posterior distribution of the location of the maximum of $f(t)$ is obtained by finding the location of the maximum of $\tilde{f}^{(g)}$ for each g. This latter computation is performed in a suitable post-processing environment such as *S* or *R*. MCMC convergence may be assessed by

standard methods (Gelman, *et al*, Section 11.6) though this remains a topic of general research interest (Fan, *et al.*, 2006).

2.2 Normal and Poisson implementations of BARS

We have implemented two versions of BARS, one using a Normal model in (3) and the other using Poisson. Our choices were based on the general interest in ordinary curve-fitting (the Normal case) and our deep and continuing interest in fitting neuronal data (the Poisson case).

The two implementations differ not only through the change of likelihood, and the resulting Laplace approximation to (4) implemented with BIC for the Poisson case, but also in the selection of starting values for the MCMC algorithm. Starting values are very important: poor choices of initial knot sets result in extremely long burn-in periods to achieve apparent stationarity. In the Poisson case we have taken advantage of the closely-related algorithm LOGSPLINE (see Hansen and Kooperberg, 2002) and have incorporated Charles Kooperberg's C implementation of it for density estimation.

Our ability to use Kooperberg's implementation for density estimation rests on the duality of fitting Poisson process intensity functions and fitting probability densities: the inhomogeneous Poisson likelihood for an intensity function $\lambda = \lambda(t)$ based on a sequence of event times $t_1, t_2, \ldots, t_n$ in an interval $(0, T]$ is

$$\begin{aligned} L(\lambda) &= p(t_1, \ldots, t_n) \\ &= e^{-\int_0^T \lambda(u)du} \prod_{j=1}^{n} \lambda(t_j). \end{aligned}$$

Here the number of events N is a Poisson random variable with expectation $\int_0^T \lambda(u)du$. Conditionally on the number of events $N = n$ the probability density becomes

$$p(t_1, \ldots, t_N | N = n) = \prod_{j=1}^{n} \lambda(t_j).$$

If we set

$$\psi(t) = \frac{\lambda(t)}{\int_0^T \lambda(u)du}$$

then it becomes clear that estimation of $\lambda(t)$ amounts to estimation of the probability density $\psi(t)$, together with estimation of $\int \lambda(u)du$. We use $N = n$ as an estimate of $\int \lambda(u)du$, and apply LOGSPLINE to estimate $\psi(t)$. LOGSPLINE returns a set of knots for a cubic spline, and these are used as initial values for BARS in the Poisson case.

LOGSPLINE is a carefully-implemented version of forward knot addition followed by backward elimination of knots. In the Normal case we have simply begun with a large set

of knots (defaulted to equally-spaced in the current version) and then performed backward elimination using BIC, until BIC fails to increase as knots are eliminated. Because LOGSPLINE is itself an effective algorithm, we believe the Poisson version of our code is likely to be more efficient than the Normal version in the sense that short MCMC run lengths can be used with Poisson. We have set the defaults for Poisson at 500 burn-in iterations and 2000 run iterations, while for the Normal these have been set to 5,000 and 20,000.

We describe below the code and wrappers for the Poisson versions. The Normal versions omit statements involving LOGSPLINE, and they also omit the regression step used to approximate the integral in (4). In describing the Poisson regressions we take the Poisson parameter to be μ. Thus, at time points $t_1, \ldots, t_n$ we have corresponding mean values $\mu_1, \ldots, \mu_n$. (In the neuronal setting with m repeated trials, for the histogram bin centered at t_j, the expected number of spiking events is $\mu_j = mw\lambda(t_j)$ where w is the bin width in seconds and $\lambda(t)$ is in units of spiking events per second.)

3 Overview of Code

Our Normal and Poisson BARS implementations proceed through the following general steps (See Figure 2):

1. Read user-defined parameters.
2. Read data, and normalize function argument (e.g., time) to interval (0,1).
3. Find initial knot set.
4. Run MCMC. For $g = 1, \ldots, G_b$, where G_b is the number of burn-in iterations, do steps (a) and (b) only; subsequently do all of (a)-(e):
   (a) Take knot step: addition, deletion, or relocation. This produces $\xi^{(g)}$.
   (b) Evaluate integral in (4), exactly in Normal case, approximately in Poisson case.
   (c) Generate $\beta^{(g)}$.
   (d) Using $\beta^{(g)}$, obtain fits and also BIC, loglikelihood, maximum, location of the maximum, and number of interior knots.
   (e) Update modal knot set (if appropriate).
5. Obtain mean and modal estimates of function values, both on a grid and at all observed argument values, and of the maximum and location of maximum.
6. Obtain confidence intervals for the maximum and location of maximum.

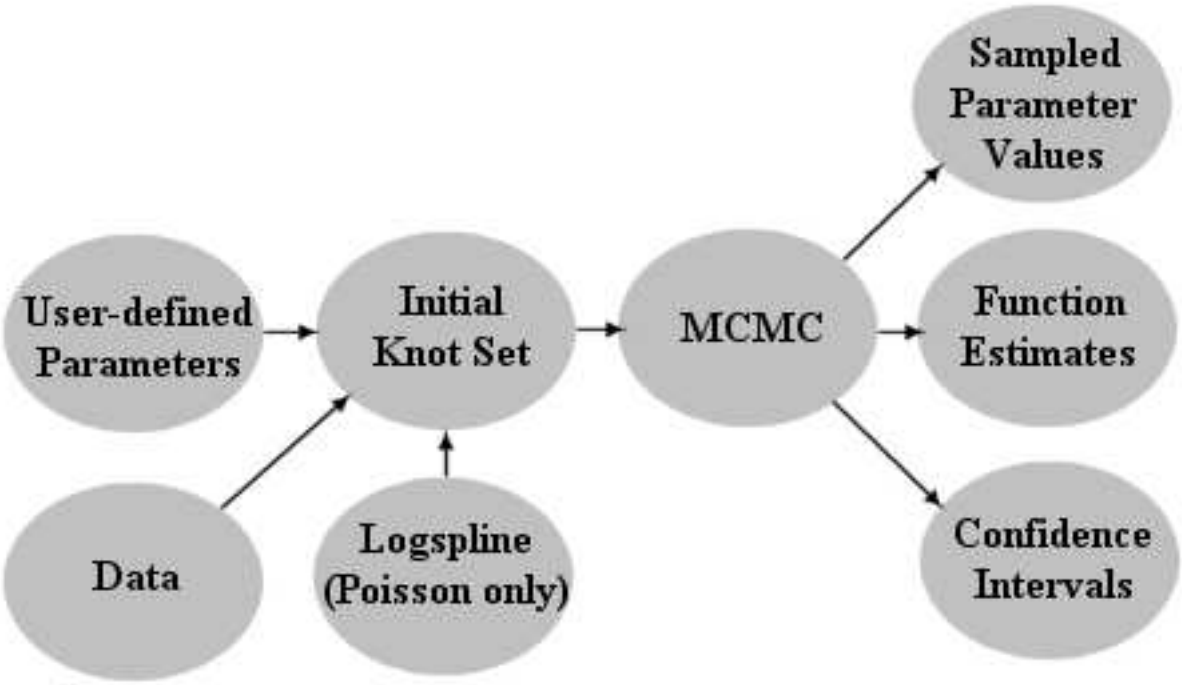

Figure 2: *Diagram of our Normal and Poisson BARS implementations. User-defined parameters and data are input and used to create the initial knot set. In the Poisson case, LOGSPLINE may be used to create the initial knot set. The MCMC runs starting from the initial knot set and outputs sampled parameter values, function estimates and confidence intervals.*

7. Write the results.

We next elaboarate on each of these steps. At the end of this section we discuss briefly the additional publicly-available subroutines that are used by BARS.

3.1 User-defined parameters

The user may set various parameters by specifying their values in the optional parameters file. The following parameters are allowed.

```
burn-in_iterations
  The number of burn-in MCMC iterations.
  Default value = 0.

sample_iterations
  The number of sample MCMC iterations
  Default value = 2000.

Use_Logspline
  "true" or "false" to indicate whether Logspline is used for
  the initial knots. If Logspline is not used, evenly spaced knots are used.
  Default value = true.

initial_number_of_knots
  The number of initial interior knots used. This only
  has an affect if Logspline is not used.
  Default value = 3.

beta_iterations
  Number of iterations for the independence chain on beta for a
  particular set of knots. It only runs a chain if the initial
  beta variate is suspect. In this case, the independence chain is
  run starting from the mle and only the last variate is used. If
  no beta candidates were accepted, no beta is used although the
  knot set is not rejected.
  Default value = 3.

beta_threshhold
  Threshhold for determining whether the initial beta variate is
  suspect. It is suspect if the log acceptance probability is
```

```
  less than the threshhold.
  Default value = -10.0.

prior_form
  This is the prior on the number of knots k.
  Possible values: Uniform, Poisson, User (for user-defined).
  Default value = Uniform.
  Uniform refers to Uniform on the interval  (L,U).
  For User, see discussion following list of parameters.

  related parameter: Uniform_parameter_L
    Default value = 1.
  related parameter: Uniform_parameter_U
    Default value = MAXKNOTS = 60.
  related parameter: Poisson_parameter_lambda
    Default value = 6.0.

  For k ~ Uniform(L,..,U)
    SET prior_form = Uniform
    SET Uniform_parameter_L = L
    SET Uniform_parameter_U = U

  For k ~ Poisson(M)
    SET prior_form = Poisson
    SET Poisson_parameter_lambda = M

proposal_parameter_tau
  Parameter that controls the spread for the knot proposal
  distribution. A candidate knot is generated by first
  selecting a current knot t, 0 < t < 1, and then selecting
  candidate knot ~ beta(t * tau, (1-t) * tau).
  Default value = 50.

reversible_jump_constant_c
  Parameter that controls the probability of
  birth and death candidates. It may be at most
  0.5.  Larger values increase the probability for birth
  death candidates. Smaller values increase the probability
  for knot relocation.
  Default value = 0.40.

confidence_level
```

```
  The confidence level for parameter intervals.
  Default value = 0.95.

number_of_grid_points
  The number of evenly spaced points for the grid along which
  fitted values will be obtained.
  Default value = 500.

verbose
  "true" or "false" to indicate verbose output.
  Default value = false.
```

User-Defined Prior on Number of Knots

The prior on the number of knots k is defaulted to Uniform. However, as discussed by DiMatteo *et al.*, Hansen and Kooperberg (2002), and Kass and Wallstrom (2002), this prior can serve usefully to control fitting. A user may specify the prior by specifying its pdf values on a set of positive integers. This is done in a file. For example, the file

```
2 0.05
3 0.15
4 0.30
5 0.30
6 0.10
7 0.10
```

indicates that $Pr(k = 2) = 0.05, \ldots, Pr(k = 7) = 0.10$. All zero probabilities that appear between the smallest k having non-zero probability and the largest k having non-zero probability are changed to an $\varepsilon > 0$. The probabilities are then scaled to sum to one. Note that very low prior probabilites may make it difficult for the chain to explore some posterior regions that have high posterior probability. For example, consider $Pr(k = 2) = Pr(k = 4) = 0.499999, Pr(k = 3) = 0.000002$. The resulting chain would generally get stuck in either the $k = 2$ region of the posterior, or the $k = 4$ region of the posterior.

3.2 Read data

The data take the form of (x, y) pairs where $x = t$ in the notation of Section 2.

3.3 Find initial knot set

By default, as discussed in Section 2, in the Poisson case the initial knot set is found using LOGSPLINE. In the Normal case evenly-spaced knots are placed and then backward elimination is used. Currently, an option to use evenly-spaced knots is also included for the Poisson case. (A planned improvement is to use every k-th value of t_j in the data for both cases.)

3.4 Run MCMC

(a) It is important to recognize that the Markov chain is on the knot sets ξ (after exact or approximate integration of (4)). As described by DiMatteo *et al.* the algorithm randomly selects as a proposal (in the Metropolis sense) either a new knot (a "birth" step), removal of a knot (a "death" step), or a relocation of a knot. Birth and relocation steps begin by randomly selecting a knot, with equal probabilities, from among all knots in the current knot set. If the proposal involves a birth step, then a location for the potential new knot is randomly selected by first randomly selecting an existing knot and then drawing from a Beta distribution centered at that knot. This Beta distribution is typically quite tight around its center (with spread controlled by an optional user-defined parameter τ) in order to propose knots that are close to existing knots (as mentioned in Section 2). The same Beta distribution is used to propose a knot relocation (the distribution being centered on the knot to be potentially relocated). As usual, the proposal to add, delete, or relocate is evaluated, and possibly accepted, via the Metropolis-Hastings ratio.

(b) Approximation of the integral in (4), in the Poisson case, is accomplished via BIC. This is obtained from the Poisson regression based on the proposed set of knots. This regression also produces the MLE and observed information matrix, needed in (c).

(c) In the Normal case, $\beta_\xi^{(g)}$ is produced by a draw from the relevant multivariate Normal distribution of the posterior of β_ξ conditionally on $\xi = \xi^{(g)}$, obtained analytically. In the Poisson case, for sufficiently large samples, the Normal approximation to the posterior based on the MLE and observed information may be used. The algorithm always attempts to draw $\beta_\xi^{(g)}$ from this approximating Normal distribution. However, if the acceptance ratio is extremely small, a short MCMC run is used instead. We have found this to be quite important because in many of our applications a Poisson mean at some time t may be very small and thus may produce a highly skewed posterior distribution.

(d) After $\beta_\xi^{(g)}$ is drawn from the conditional posterior, fits may be obtained. In our applications we are often interested in maxima and their locations. Therefore, by default, we compute these.

(e) If BIC for the newly-obtained, current $\xi^{(g)}$ is larger than BIC for all preceding knot sets then the current $\xi^{(g)}$ is retained as the current modal knot set. The MLE $\hat{\beta}_\xi^{(g)}$ (in the Normal case, the least-squares estimate) is also retained so that modal fits can be produced for the final modal knot set.

3.5 Obtain estimates

Ater MCMC terminates, the iterations are used to find posterior mean and modal fits. The fits are obtained both at all t_j values and on a user-specified grid. In addition, the mean and modal values of the function maximum and its location are computed.

3.6 Obtain posterior intervals

Based on the set of draws from the posteriors that are used to form estimates, quantiles are computed via partial sorting. By default the .025 and .975 quantiles form the 95% posterior ("credible") intervals. Currently, confidence intervals are only obtained for the function maximum and location of the maximum. Other intervals may be obtained easily from the draws from the posterior in the output.

3.7 Write results

Results are written into a series of files:

```
sampled_knots_file
  The name of the output file for the sampled knot locations.
  Format is
  [iteration number] [number of interior knots] [knots...]
  Note that the number of entries per line will vary.
  Use "none" to indicate no file.
  Default value = none.

sampled_mu_file
```

```
  The name of the output file for sampled fitted values (mu)
  at the observed argument points t_1, ..., t_n.
  Use "none" to indicate no file.
  Default value = samp_mu.

sampled_mu-grid_file
  The name of the output file for sampled fitted values (mu)
  on the evenly spaced grid points.
  Use "none" to indicate no file.
  Default value = none.

sampled_params_file
  The name of the output file for sampled values of
  various parameters. The parameters given are
  iteration number, BIC, log likelihood, location of peak, height at
  peak, number of interior knots.
  Use "none" to indicate no file.
  Default value = samp_params.

summary_mu_file
  The name of the output file for summary fitted values (mu)
  at the observed argument points t_1, ..., t_n.
  Three rows are given with
  the t values, posterior mean, and posterior mode,
  respectively.
  Use "none" to indicate no file.
  Default value = summ_mu.

summary_mu-grid_file
  The name of the output file for summary fitted values (mu)
  on the evenly spaced grid points. Three rows are given with
  the t values, posterior mean, and posterior mode,
  respectively.
  Use "none" to indicate no file.
  Default value = none.

summary_params_file
  The name of the output file for summary values of
  certain parameters. The first row is for the location of the
  peak, and the second is for the height, and the third is for the
  number of interior knots. The first two columns are quantiles giving
  a confidence interval for the specified confidence level (see
```

```
parameter confidence_level). The third column is the posterior mean,
and the fourth is the posterior mode.
Use "none" to indicate no file.
Default value = summ_params.
```

3.8 External Subroutines for BARS

In addition to incorporating the LOGSPLINE code, we have used routines for manipulating B-Splines written by Bates and Venables (included in the release of R, 2003) and for random number generation obtained from Ranlib (Brown and Lovato, 1996). These are part of the BARS code.

BARS also calls linear algebra subroutines from LAPACK and BLAS. These are available at
`http://www.netlib.org/lapack` and `http://www.netlib.org/blas`. They must be installed prior to compiling BARS. Information about these packages may be found in Anderson *et al.* (1999) and Dongarra *et al.* (1990a,b).

4 R and S Wrappers

Prior to using this wrapper, one must build a CHAPTER using the file barsP.c. The shared library must be made available with the dyn.open("S.so") command. Also, the barsP.c program must be properly compiled, with the compiled program saved as barsP.out.

Note that due to the large amount of output generated by the program, it is desirable to save the results of the program into a variable, such as:

```
out_barsP.fun(x,y,.....)
```

The wrapper performs the operation by reading the data and settings into the files

```
bars_points
bars_params
```

Function call

```
barsP.fun(x,y,initial,iknots,prior,priorparam,burnin,sims,
        tau,c,fits,peak,conf,trials,bins)
```

Required Input

```
x
    a vector of the independent variable, in increasing order
y
    a vector of the dependent variable, in the form of count data
```

Optional Input

```
Initial knots settings:
  initial
    use logspline ("logspline") or evenly spaced ("even" or "equal")
    for initial knots (default is "logspline")
  iknots
    the initial number of knots for the spline if using evenly
    spaced knots (default = 3)

Settings on prior for knots:
  prior
    the type of prior being used for the knots (the only acceptable
      answers are "Poisson", "uniform", and "user"  - default = "uniform")
  priorparam
    the parameter for the prior
    a) if using Poisson, the choice for lambda = mean
    b) if using Uniform, a vector of length 2 which includes the
    minimum number of knots followed by the maximum number of
    knots  (default = c(1,60))
    c) if using user-defined prior, a matrix with 2 columns.  The
    first column should be the number of knots and the second column
    should be the probability of obtaining this number of knots.

MCMC settings:

  burnin
    the desired length of the burn-in for the MCMC chain
    (default = 200)
  sims
    the number of simulations desired for the MCMC chain
    (default = 2000)
  tau
    parameter that controls the spread for the knot proposal
```

```
    distribution (default = 50.0)
  c
    parameter that controls the probability of birth and death
    candidates (default = 0.4)

Output settings:

  fits
    if "T", the program will return the fitted values for each
    x-value for each run of the simulation (default = T)  Please
    note that if the number of data points and/or simulations is
    large, there may be a lengthy delay as the necessary data is
    read.
  peak
    if "T", the program will return the location and height of the
    highest point on the fitted curve (default = F)
  conf
    for use with peak.  Sets the probability for the credible intervals
    for the location and height of the peak.  (default = 0.95 for 95%
    credible intervals).

Other settings:
  trials
    the number of trials that are concatenated in the data
    (default = 1 )
  bins
    the number of bins one desires the x-axis to be divided into -
    used to handle unbinned data and calculate posterior modes
    (default = 150)
```

Output

```
postmeans
   vector of the posterior means evaluated at the x values
postmodes
   vector of the posterior modes evaluated at the x values
sims
   vector of each simulation number, beginning at burnin + 1 and
      ending at burnin + sims
no.knots
   vector of the number of knots used at each trial (does not include
```

```
      burnin iterations)
sampknots
   matrix containing the position of the knots at each iteration.
      Length of the matrix is equal to the number of iterations, not
      including burnin iterations, with the width of the matrix equal to
      the maximum number of knots at any iteration.  NAs are used to
      fill in the matrix at iteration numbers that have less than the
      maximum number of knots.
sampBICs
   vector of the calculated BIC at each trial (does not include burnin
      iterations)
sampllikes
   vector of the calculated loglikelihood at each trial (does not
      include burnin iterations)
```

Optional Output

```
Optional output if fits = "T":
  sampfits
    matrix of fits for each trial (does not include burnin
    iterations), with the rows of the matrix corresponding to the
    individual trial.  The columns represent the fits at each value
    of x.

Optional output if peak = "T":
  samplpeaks
    vector of the x location of the highest point in the fitted
    curve for each trial (does not include burnin iterations)
  samphpeaks
    vector of the y value (height) of the highest point in the
    fitted curve for each trial (does not include burnin iterations)
  peaklocationquantile
    a credible interval for the x location of the highest peak; width
    of the interval is dependent upon the setting chosen for conf
  peaklocationmean
    the mean x location for the highest peak
  peaklocationmode
    the mode x location for the highest peak
  peakheightquantile
    a credible interval for the y value (height) of the highest peak;
    width of the interval is dependent upon the setting chosen for
```

```
  conf
peakheightmean
  the mean y value (height) of the highest peak
peakheightmode
  the mode y value (height) of the highest peak
```

5 Pseudo-Code

5.1 Function: BARS for Poisson Count Data

- Read Data
  - **comment:** For Poisson count data, the data must take the form of pairs of bin midpoints and Poisson counts. The number of replicated data sets is also required. In neuron firing examples, this is the number of trials. In many other examples, the number of replicated data sets may be set to one.
- Read User Parameters
  - **comment:** These include parameters that specify the form of the prior, prior parameters, mcmc parameters, and parameters that specify output variables and the destination files.
- Normalize Data
  - **comment:** The bin midpoints are normalized to lie between 0 and 1, inclusive.
- MCMC
- Write summary output of desired parameters
- **return**

5.2 Function: MCMC

In the following, a **model** M_* contains information for an individual MCMC iteration. In particular, it contains

- k_*, the number of interior knots
- ξ_*, the set of interior knots

- $X_{D,*}$, the design basis, and $X_{G,*}$, the grid basis. The design basis is based on the input bin midpoints. The grid basis is based upon a grid of evenly spaced points, the number of which is user defined.

- statistical model fitting information, including parameter estimates, error estimates, and failure to fit.

- $\beta_* \sim \pi(\beta|k_*, \xi_*, Data)$

- $\mu_{D,*} = \exp(X_D \beta_*)$

- $\mu_{G,*} = \exp(X_G \beta_*)$

- The BIC and log likelihood for the full model with parameters (k_*, ξ_*, β_*).

- Declare models M_{curr}, M_{cand}, and M_{temp}.
- Set initial knots in M_{curr}. **comment:** The initial knots may be user defined, equally spaced, or obtained via logspline.
- Calculate birth and death probabilities for each possible value of k, using the user-defined prior, as follows:
  - **if** $(k >= MAXKNOTS)$
    - $birth\ probability = 0$
  - **else**
    - $birth\ probability = c \min(1, \pi(k+1)/\pi(k))$
  - **if** $(k <= 1)$
    - $death\ probability = 0$
  - **else**
    - $death\ probability = c \min(1, \pi(k-1)/\pi(k))$
  - **comment:** knot relocation occurs with probability $1 - (birth\ probability + death\ probability)$.
- **comment:** Define $\mu^{(0)}$, used to start each iterative fitting process.
- **for** $i \leftarrow 0$ **to** $(n-1)$
  - $\mu_i^{(0)} = \max(0.1, y_i)$
- Form the natural spline design basis for $M_{curr} \rightarrow X_{D,curr}$
- Fit the Poisson Regression Model for M_{curr}

- **if** (fit of M_{curr} failed)
  - Remove knots through backwards elimination until a subset is found such that 1) the model fit does not fail, 2) the model has the greatest likelihood among the models with the same number of knots in which the fit does not fail, and 3) the knot subset has positive prior probability. If all models with a given number of knots fail to be fit, the model in which $X_D^T W X_D$ has the smallest condition number is selected, and the procedure continues by trying to remove an additional knot.
  - If backwards elimination fails to find a valid subset, the procedure tries to fit a model with the minimum number of knots that have positive prior probability, with the knots equally spaced. If the model fails to be fit, **exit**.
  - Set M_{curr} to the resulting model.
- $maxBIC \leftarrow 0$
- $total\ iterations \leftarrow burnin\ iterations + sampling\ iterations$
- **for** $i \leftarrow 0$ **to** $total\ iterations - 1$
  - $u \sim U(0,1)$
  - **if** $(u < birth\ probability)$
    - $s \sim Discrete\ Uniform(\xi_{curr})$
    - $t \sim beta(\alpha = s\tau, \beta = (1-s)\tau)$
    - $\xi_{cand} \leftarrow \xi_{curr} \cup \{t\}$
    - $k_{cand} \leftarrow k_{curr} + 1$
    - Form the natural spline design basis for $M_{cand} \rightarrow X_{D,cand}$
    - Fit the Poisson Regression Model for M_{cand}
    - **if** (fit of M_{cand} failed) $accept\ probability = 0$
    - **else**
      - $dens \leftarrow q(M_{cand}|M_{curr})k_{curr} = \sum_{r \in \xi_{curr}} f_{beta}(t|\alpha = r\tau, \beta = (1-r)\tau)$
      - $accept\ probability = \exp(\ell_{cand} - \ell_{curr} + \log(k_{curr}) - \log(dens) - 0.5\log(n))$
      - **comment:** ℓ_* in the above equation is the profile likelihood, $\ell_* = \sup_\beta \ell(\xi_*, k_*, \beta)$
  - **else if** $(1 - u < death\ probability)$
    - $t \sim Discrete\ Uniform(\xi_{curr})$
    - $\xi_{cand} \leftarrow \xi_{curr} \setminus \{t\}$
    - $k_{cand} \leftarrow k_{curr} - 1$
    - Form the natural spline design basis for $M_{cand} \rightarrow X_{D,cand}$
    - Fit the Poisson Regression Model for M_{cand}

    * **if** (fit of M_{cand} failed) *accept probability* $= 0$
    * **else**
        · $dens \leftarrow q(M_{curr}|M_{cand})k_{cand} = \sum_{r \in \xi_{cand}} f_{beta}(t|\alpha = r\tau, \beta = (1-r)\tau)$
        · *accept probability* $= \exp(\ell_{cand} - \ell_{curr} - \log(k_{cand}) + \log(dens) + 0.5\log(n))$
- **else**
    * $s \sim Discrete\ Uniform(\xi_{curr})$
    * $t \sim beta(\alpha = s\tau, \beta = (1-s)\tau)$
    * $\xi_{cand} \leftarrow (\xi_{curr} \cup \{t\}) \setminus \{s\}$
    * $k_{cand} \leftarrow k_{curr}$
    * Form the natural spline design basis for $M_{cand} \rightarrow X_{D,cand}$
    * Fit the Poisson Regression Model for M_{cand}
    * **if** (fit of M_{cand} failed) *accept probability* $= 0$
    * **else**
        · $dens_1 \leftarrow q(M_{cand}|M_{curr})k_{curr} = f_{beta}(t|\alpha = s\tau, \beta = (1-s)\tau)$
        · $dens_2 \leftarrow q(M_{curr}|M_{cand})k_{cand} = f_{beta}(s|\alpha = t\tau, \beta = (1-t)\tau)$
        · *accept probability* $= \exp(\ell_{cand} - \ell_{curr} + \log(dens_1) - \log(dens_2))$
- $u \sim U(0,1)$
- **if** ($u <$ *accept probability*)
    * **comment:** Candidate model accepted. Swap M_{curr} and M_{cand}.
    * $M_{temp} \leftarrow M_{curr}$
    * $M_{curr} \leftarrow M_{cand}$
    * $M_{cand} \leftarrow M_{temp}$
- **if** ($i >=$ *burnin iterations*)
    * **comment:** Beyond the burn-in period.
    * Generate Random Coefficient Vector for M_{curr}
    * Calculate BIC_{curr} and ℓ_{curr} for the full model using parameter values $(k_{curr}, \xi_{curr}, \beta_{curr})$.
    * $\mu_{D,curr} \leftarrow \exp(X_{D,curr}\beta_{curr})$
    * Form the natural spline grid basis for $M_{curr} \rightarrow X_{G,curr}$
    * $\mu_{G,curr} \leftarrow \exp(X_{G,curr}\beta_{curr})$
    * **comment:** Find mode and the mean function evaluated at the mode. In neuron firing examples, it produces the location of the peak firing rate, and the peak firing rate.
    * Use $\mu_{G,curr}$ to form an interpolating spline.
    * Locate mode of interpolating spline $\rightarrow (x_{mode}, \mu_{curr}(x_{mode}))$
    * Write desired parameters to a file.

        * Store desired parameters for later use.
        * **comment:** update posterior modal values, if appropriate
        * **if** $((BIC_{curr} > maxBIC)$ **or** $(i == burnin\ iterations))$
            · $maxBIC \leftarrow BIC_{curr}$
            · parameter modes $\leftarrow$ current parameter values
- Use partial sorting to form confidence intervals of desired stored parameters.
- Calculate means of desired stored parameters.
- **return**

5.3 Function: Fit Poisson Regression Model

- **input:**
    - X, an $n \times p$ design matrix
    - y, a vector of observed counts
    - $\mu^{(0)}$, a vector of starting values
- **output:**
    - $\widehat{\beta}$, estimated coefficients
    - U, a $p \times p$ upper triangular matrix that contains information on the estimated covariance matrix of $\widehat{\beta}$.
    - $error$, a boolean indicator of fit failure.
- $\mu \leftarrow \mu^{(0)}$
- $j \leftarrow 0$
- $\ell_0 \leftarrow 0$
- $error \leftarrow$ **false**
- **repeat**
    - $j \leftarrow j + 1$
    - $z \leftarrow \log \mu + (y - \mu)/\mu$.
    - $W \leftarrow Diag(\mu)$
    - $H \leftarrow WX$. **comment:** Use known diagonal structure of W
    - $J \leftarrow H^T X$

- $U \leftarrow$ Upper triangular matrix from the Cholesky decomposition of $J = U^T U$
- **if** Cholesky decomposition fails
  - $error \leftarrow$ **true**
  - $exit \leftarrow$ **true**
- **else**
  - $\widehat{\beta} \leftarrow$ Solution to $J\beta = H^T z$. **comment:** Use Cholesky decomposition of J
  - **if** unable to solve equation
    - $error \leftarrow$ **true**
    - $exit \leftarrow$ **true**
  - **else**
    - $\eta \leftarrow X\widehat{\beta}$
    - $\mu \leftarrow \exp(\eta)$
    - $\ell_j \leftarrow \sum_i (y_i \eta_i - \mu_i)$
    - $exit \leftarrow (((|\ell_j - \ell_{j-1}| < \varepsilon)$ **and** $(j > 1))$ **or** $(j > 20))$

- **until** $(exit)$
- **return**

5.4 Function: Generate Random Coefficient Vector

- Generates β from the posterior distribution $\pi(\beta|k, \xi, Data)$ or from the $N\left(\widehat{\beta}, \left(X^T W X\right)^{-1}\right)$ approximation, as follows. One β is generated from the normal approximation. If the β appears to *not* be an outlier under the posterior distribution, the β variate is accepted and returned. If the β does appear to be an outlier, the routine takes a user-defined number of Metropolis-Hastings steps and returns the last sampled β. The β is identified as an outlier if its log Metropolis-Hastings acceptance probability is below a user-defined threshhold.
- Let π^* denote the density for the normal approximation. Note that the Cholesky decomposition of $X^T W X$ is already available from the last fitting iteration.

- **input:**
  - $\widehat{\beta}$, the MLE of β.
  - U, a $p \times p$ upper triangular matrix that contains information on the estimated covariance matrix of $\widehat{\beta}$.

  - MHI, the number of Metropolis-Hastings iterations. This is a user-defined parameter.
  - MHT, the threshhold used to determine whether (a) the initial β variate from the normal approximation should be kept as the resulting variate, or (b) MHI Metropolis-Hastings iterations are used. MHT is compared to the log of the acceptance probability. MHT is a user-defined parameter.

- **output:**
  - $error$, a boolean indicator of failure
  - β, the random variate. Only defined if **not** $error$.
- $\beta_{curr} \leftarrow \widehat{\beta}$
- **for** $i \leftarrow 0$ **to** $MHI - 1$
  - $iter \leftarrow 0$, $error \leftarrow$ **false**, $exit \leftarrow$ **false**
  - **repeat**
    - $iter \leftarrow iter + 1$
    - $z \sim N(0, I)$
    - $A \leftarrow$ Solution to $U^T A = z$.
    - $error \leftarrow$ (unable to solve equation)
    - $exit \leftarrow$ ((**not** $error$) **or** ($iter \geq 20$))
  - **until** ($exit$)
  - **if** ($error$) **exit**
  - **else**
    - $\beta_{cand} \leftarrow \widehat{\beta} + A$
    - $r \leftarrow \log\left(\frac{L(k,\xi,\beta_{cand})}{L(k,\xi,\beta_{curr})} \frac{\pi(\beta_{cand}|k,\xi)}{\pi(\beta_{curr}|k,\xi)} \frac{\pi^*(\beta_{curr}|k,\xi,Data)}{\pi^*(\beta_{cand}|k,\xi,Data)}\right)$
    - **if** (($i = 0$) **and** ($r > MHT$))
      - **comment**: Accept the initial variate. No additional Metropolis-Hastings steps.
      - $i \leftarrow MHI$
      - $u \leftarrow r - 1.0$
    - **else**
      - $u \leftarrow U(0, 1)$
      - $u \leftarrow \log(u)$
    - **if** ($u < r$)
      - **comment**: Accept the candidate β.
      - $\beta_{curr} \leftarrow \beta_{cand}$

- $beta \leftarrow \beta_{curr}$.
- **return**

6 Discussion

The implementation of BARS we have described here improves on the original code by being more reliable and very much faster. There are versions not only for the standard nonparametric regression in Equation (1) but also for the Poisson regression setting Equation (2), which is of great interest in neurophysiological applications. The code is self-contained in that it includes likelihood maximization for the Poisson regression problem. An important feature of the implementation in this setting is its incorporation of LOGSPLINE code, which provides initial values for BARS and thereby greatly reduces the number of MCMC iterations required. Based on experimentation with neuronal data we determined our defaults of 500 burn-in iterations and 2000 MCMC iterations in the Poisson case. For typical data sets, using 2GHz computers, BARS takes only a few seconds to run.

There are some noteworthy limitations of our current implementation. First, we rely on Poisson regression even for sparse data where the counts are either 0 or 1. This produces a large amount of unnecessary computation: a better approach, in that setting, would be to use the Poisson process likelihood function. It would be nice to have a version of the code that specifically treats the sparse case. Second, our code does not allow a user-defined prior on knot locations, which would be helpful when there is a suspicion that the function may be varying rapidly in a specified part of its domain. Finally, the only immediately-available priors on the number of knots k are uniform (the default) and Poisson. The code does, however, allow the user to input a different prior (see the end of Section 3.1, above). To be specific, we have noted that BARS uses BIC as a model-weighing criterion. One obvious alternative would be to use AIC or one of its generalizations, as in the general form in Hansen and Kooperberg (2002). As discussed in Kass and Wallstrom (2002), there is a relatively simple interpretation when the $\log n$ factor in BIC is replaced by $\log c$ for any postive constant c. In our implementation, this suggests a user-defined prior of the form

$$p(k) \propto (\frac{n}{c})^{\frac{k}{2}}$$

where $p(k)$ is the probability assigned to k. Suitable choices of $c < n$ will tend to drive BARS toward decreased smoothness. Such priors are easy to introduce using the current code, but they do require a preliminary calculation of $p(k)$ over values of k for which $p(k)$ is non-negligible.

A concern in neurophysiology is that important information may be lost when data are

pooled across trials. On the one hand, because the PSTH (the histogram of Figure 1) is based on counts of firing events after aggregrating across trials, it estimates a trial-averaged (marginal) firing intensity and is, thus, usually taken to represent a stimulus-related response beyond any trial-dependent "noise." BARS offers a great improvement over the PSTH and, we believe, on other available methods of estimating this marginal firing intensity. On the other hand, within-trial effects are crucial in many contexts. A procedure that begins with BARS, but then also takes into account trial-to-trial variability and non-Poisson neuronal firing, as well as certain types of oscillatory effects, is part of the Ph.D. thesis work of J. Liebner, under the supervison of R. Kass, and will be reported elsewhere.

References

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999) *LAPACK Users' Guide, 3rd Ed.*, Society for Industrial and Applied Mathematics, Philadelphia, PA.

Baker, C.I., Behrmann, M., and Olson, C.R. (2002) Impact of learning on representation of parts and wholes in monkey inferotemporal cortex. *Nature Neurosci.*, 5: 1210-1216.

Behseta, S. and Kass, R.E. (2005) Testing Equality of Two Functions using BARS, *Statistics in Medicine,* 24:3523-34.

Behseta, S., Kass, R.E., and Wallstrom, G.L. (2005) Hierarchical models for assessing variability among functions, *Biometrika,* 92: 419-434.

Behseta, S., Kass, R.E., Moorman, D. and Olson, C. (2007) Testing Equality of Several Functions: Analysis of Single-Unit Firing Rate Curves Across Multiple Experimental Conditions, *Statistics in Medicine*, to appear.

Brown, B.W. and Lovato, J. (1996) Library of C routines for random number generation. `http://www.stat.umn.edu/HELP/ranlib-docs/ranlib.c.chs`

Denison, D. G. T., Mallick, B. K. and Smith, A. F. M. (1998). Bayesian MARS, Statist. Comp. 8: 337-346.

DiMatteo, I., Genovese, C.R., and Kass, R.E. (2001) Bayesian curve fitting with free–knot splines. *Biometrika*, 88: 1055-1073.

Dongarra, J.J., Du Croz, J., Duff, I.S., and Hammarling, S. (1990a) A set of level 3 basic linear algebra subprograms, *ACM Trans. Math. Soft.*, 16: 1–17.

Dongarra, J.J., Du Croz, J., Duff, I.S., and Hammarling, S. (1990b) Algorithm 679: A set of level 3 basic linear algebra subprograms, *ACM Trans. Math. Soft.*, 16: 18–28.

Fan, Y., Brooks, P., and Gelman, A. (2006) Output assessment for Monte Carlo simulations via the score statistic. *Journal of Computational and Graphical Statistics*, 15: 178-206.

Gelman, A., Carlin, J., Stern, H., and Rubin, D. (2004) Bayesian Data Analysis, Second Edition, Chapman & Hall/CRC, Boca Raton.

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

Kass, R.E., Ventura, V. and Cai, C. (2003) Statistical smoothing of neuronal data. *Network: Comput. Neural Systems*, 14: 5-15.

Kass, R.E. and Wallstrom, G. Invited comment on "Spline Adaptation in Extended Linear Models" by Mark H. Hansen and Charles Kooperberg, *Statistical Science*, *Statistical Science*, 17: 24-29.

Liu, J.S., Wong, W.H., and Kong, A. (1994) Covariance Structure of the Gibbs Sampler with Applications to the Comparisons of Estimators and Augmentation Schemes, *Biometrika*, 81: 27-40.

Pauler, D.K. (1998) The Schwarz criterion and related methods for normal linear models, *Biometrika*, 85: 13-27.

R Development Core Team. (2003) R: A language and environment for statistical computing. R Foundation for Statistical Computing.
`http://www.R-project.org`.

Stone, C.J., Hansen M., Kooperberg, C., and Truong, Y.K. The use of polynomial splines and their tensor products in extended linear modeling (with discussion) (1997). *Annals of Statistics*, 25: 13711470.

Venables, W.N. and Ripley, B.D. (1999) Modern Applied Statistics with S, Third Edition, Springer.

Wallstrom, G.L., Kass, R.E., Miller, A., Cohn, J.F., and Fox, N.A. (2004). Automatic correction of ocular artifact in the EEG: a comparison of regression-based and component-based methods. *International Journal of Psychophysiology*, 53: 105-119.

Zhang, Xiaohua, Roeder, Kathryn, Wallstrom, Garrick, Devlin, B. (2003) Integration of association statistics over genomic regions using Bayesian Adaptive Regression Splines. *Human Genomics*, 1: 20-29.