BayesBD: An R Package for Bayesian Inference on Image Boundaries

by Nicholas Syring and Meng Li

Abstract

We present the **BayesBD** package providing Bayesian inference for boundaries of noisy images. The **BayesBD** package implements flexible Gaussian process priors indexed by the circle to recover the boundary in a binary or Gaussian noised image. The boundary recovered by **BayesBD** has the practical advantages of guaranteed geometric restrictions and convenient joint inferences under certain assumptions, in addition to its desirable theoretical property of achieving (nearly) minimax optimal rate in a way that is adaptive to the unknown smoothness. The core sampling tasks for our model have linear complexity, and are implemented in C++ for computational efficiency using packages **Rcpp** and **RcppArmadillo**. Users can access the full functionality of the package in both the command line and the corresponding **shiny** application. Additionally, the package includes numerous utility functions to aid users in data preparation and analysis of results. We compare **BayesBD** with selected existing packages using both simulations and real data applications, demonstrating the excellent performance and flexibility of **BayesBD** even when the observation contains complicated structural information that may violate its assumptions.

Introduction

Boundary estimation is an important problem in image analysis with wide-ranging applications from identifying tumors in medical images (Li et al., 2010), classifying the process of machine wear by analyzing the boundary between normal and worn materials (Yuan et al., 2016), to identifying regions of interest in satellite images, such as the boundary of Scotland's Lake Menteith (Cucala and Marin, 2014; Marin and Robert, 2014). Furthermore, boundaries present in epidemiological or ecological data may reflect the progression of a disease or an invasive species; see Waller and Gotway (2004), Lu and Carlin (2005), and Fitzpatrick et al. (2010).

There is a rich literature on image segmentation for both noise-free and noisy observations; see the surveys in Ziou and Tabbone (1998); Basu (2002); Maini and Aggarwal (2009); Bhardwaj and Mittal (2012), and particularly the Bayesian approaches in Hurn et al. (2003) and Grenander and Miller (2007). Recently, Li and Ghosal (2015) developed a flexible nonparametric Bayesian model to detect image boundaries, which achieved four aims of guaranteed geometric restriction, (nearly) minimax optimal rate adaptive to the smoothness level, convenience for joint inference, and computational efficiency. However, despite the theoretical soundness, the practical implementation of Li and Ghosal's method is far from trivial, mostly in the approachability of the proposed nonparametric Bayesian framework and further improvement in the speed of posterior sampling algorithms, which becomes critical in attempts to popularize this approach in statistics and the broader scientific community. In this paper, we present the R package **BayesBD** (Syring and Li, 2017) which aims to fill this gap. The developed **BayesBD** package provides support for analyzing binary images and Gaussian-noised images, which commonly arise in many applications. We implement various options for posterior calculation including the Metropolis-Hastings sampler (Hastings, 1970) and slice sampler (Neal, 2003). To further speed up the Markov Chain Monte Carlo (MCMC), we take advantage of the integration via **RcppArmadillo** (Eddelbuettel, 2013; Eddelbuettel and Sanderson, 2014) of R and the compiled C++ language. We further integrate the **BayesBD** package with **shiny** (RStudio, Inc, 2016) to facilitate the usage of implemented boundary detection methods in real applications.

A far as we know, there are no other R packages for image boundary detection problems achieving the four goals mentioned above. An earlier version of the **BayesBD** package (Li, 2015) provided first-of-its-kind tools for analyzing images, but support for Gaussian-noised images, C++ implementations, more choices of posterior samplers, and **shiny** integrations were not available until the current version. For example, the nested loops required for MCMC sampling were inefficient in R programming. The combination of new programming and faster sampling algorithms means that a typical simulation example consisting of 5000 posterior samples from $10,000$ data points can now be completed in about one minute.

The rest of the paper is organized as follows. We first introduce the problem of statistical inference on boundaries of noisy images, the nonparametric Bayesian models in use, and posterior sampling algorithms. We then demonstrate how to use the main functions of the package for data analysis working with both the command line and **shiny**. We next conduct a comprehensive experiment on the comparison of sampling methods and coding platforms, scalability of **BayesBD**, and comparisons

with existing packages including **mritc**, **bayesImageS**, and **bayess**. We illustrate a pair of real data analyses of medical and satellite images. The paper is concluded by a Summary section.

Statistical analysis of image boundaries

Image data

An image observed with noise may be represented by a set of data points or pixels $(Y_i, X_i)_{i=1}^n$, where the intensities Y_i are observed at locations $X_i \in \mathcal{X} = [0,1]^2$. Following Li and Ghosal (2015), we assume that there is a closed region $\Gamma \in \mathcal{X}$ such that the intensities Y_i are distributed as follows conditionally on whether its location is inside Γ or in the background Γ^c:

$$Y_i \sim \begin{cases} f(\cdot;\xi) & X_i \in \Gamma; \\ f(\cdot;\rho) & X_i \in \Gamma^c, \end{cases} \tag{1}$$

where f is a given probability mass function or probability density function of a parametric family up to unknown parameters (ξ, ρ). For example, Figure 1 shows two simulated images where the parametric family is Bernoulli and Gaussian, respectively. These images can be reproduced using the functions `par2obs`, `parnormobs`, and `plotBD` which will be demonstrated in detail later on.

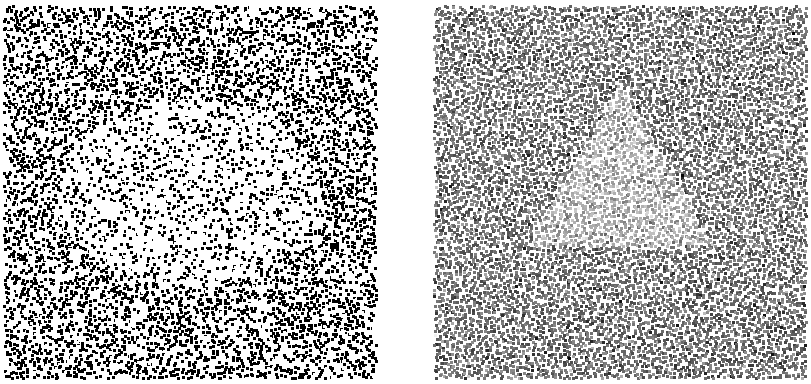


Figure 1: Left: a binary image generated using an elliptical boundary and parameters $\pi_1 = 0.65$ and $\pi_2 = 0.35$. Right: a Gaussian-noised image generated using a triangular boundary and parameters $\mu_1 = 1$, $\mu_2 = -1$, and $\sigma_1 = \sigma_2 = 1$. Both images have the size 100×100.

The parameter of interest is the boundary of the closed region $\gamma := \partial\Gamma$, which is assumed to be closed and smooth, while (ξ, ρ) are nuisance parameters. We make the following assumptions about the noisy image:

1. The pixel locations X_i are sampled either completely randomly, i.e., $X_i \overset{i.i.d.}{\sim} \mathsf{Unif}(\mathcal{X})$ or jitteredly randomly, i.e., $\mathcal{X}$ is first partitioned into blocks $\mathcal{X}_i$ using an equally-spaced grid and then locations are sampled $X_i \sim \mathsf{Unif}(\mathcal{X}_i)$.
2. The closed region Γ is star-shaped with a known reference point $O \in \Gamma$, i.e., the line segment joining O to any point in Γ is also in Γ.
3. The boundary γ is an $\alpha-$Hölder smooth function, i.e., $\gamma \in \mathbb{C}^\alpha(\mathbb{S})$ where

$$\mathbb{C}^\alpha(\mathbb{S}) := \{f : \mathbb{S} \to \mathbb{R}^+, |f^{(\alpha_0)}(x) - f^{(\alpha_0)}(y)| \leq L_f \|x - y\|^{\alpha - \alpha_0}, \forall x, y \in \mathbb{S}\}$$

, where $\mathbb{S}$ is the unit circle, α_0 is the largest integer strictly smaller than α, L_f is some positive constant, and $\|\cdot\|$ is the Euclidean distance.

Assumptions 2 and 3 imply that the region of interest is star-shaped with a smooth boundary. While these assumptions are crucial to guarantee desirable asymptotic properties of the estimator implemented in **BayesBD** and are reasonable in many applications, it is certainly of great interest to investigate the performance of **BayesBD** when these assumptions are violated. In what follows, we study numerous examples that are not uncommon in practice but violate these assumptions to some extent, to demonstrate the flexibility of **BayesBD** and its capacity to handle practical images that may be much more complicated than the two-region setting with assumptions above. These examples include the triangular boundary in Figure 1 which has a piecewise smooth boundary. and

thus violates Assumption 3, and three real data examples including the image of Lake Menteith 8 and two neuroimaging examples 6 where multiple regions are present and the region of interest has non-smooth or even discontinuous boundary.

Letting Θ be the parameter space of the parametric family f, conditions to separate the inside and outside parameters are needed. Examples of the parameter space Θ^* for (ξ, ρ) include but are not limited to:

4A. One-parameter family such as Bernoulli, Poisson, exponential distributions, and $\Theta^* = \Theta^2 \cap \{(\xi, \rho) : \rho < \xi\}$, or $\Theta^* = \Theta^2 \cap \{(\xi, \rho) : \rho > \xi\}$.

4B. Two-parameter family such as Gaussian distributions, and $\Theta^* = \Theta^2 \cap \{((\mu_1, \sigma_1), (\mu_2, \sigma_2)) : \mu_1 > \mu_2, \sigma_1 = \sigma_2\}$, or $\Theta^* = \Theta^2 \cap \{((\mu_1, \sigma_1), (\mu_2, \sigma_2)) : \mu_1 > \mu_2, \sigma_1 > \sigma_2\}$, or $\Theta^* = \Theta^2 \cap \{((\mu_1, \sigma_1), (\mu_2, \sigma_2)) : \mu_1 = \mu_2, \sigma_1 > \sigma_2\}$.

In practice, the order restriction in 4A or 4B is often naturally obtained depending on the concrete problem. For instance, in brain oncology, a tumor often has higher intensity values than its surroundings in a positron emission tomography scan, while for astronomical applications objects of interest emit light and will be brighter. A more general condition for any parametric family can be referred to Condition (C) in Li and Ghosal (2015).

It is worth noticing that although model 1 follows a two-region framework, the method in Li and Ghosal (2015) and our developed **BayesBD** have the flexibility to handle data with multiple regions by running the two-region method iteratively, which is demonstrated in the neuroimaging application below.

A nonparametric Bayesian model for image boundaries

Let $Y = \{Y_i\}_{i=1}^n$ and $X = \{X_i\}_{i=1}^n$, then the likelihood of the image data described in (1) is

$$L(Y|X, \theta) = \prod_{i \in I_1} f(Y_i; \xi) \prod_{i \in I_2} f(Y_i; \rho),$$

where $I_1 = \{i : X_i \in \Gamma\}$, $I_2 = \{i : X_i \in \Gamma^c\}$, and θ denotes the full parameter (ξ, ρ, γ).

We view γ as a curve mapping $[0, 2\pi] \to \mathbb{R}^+$ and model it using a randomly rescaled Gaussian process prior on the circle $\mathbb{S}$: $\gamma(\omega) \sim \mathsf{GP}(\mu(\omega), G_a(\cdot, \cdot)/\tau)$ where the covariance kernel

$$\begin{aligned} G_a(t_1, t_2) &= \exp(-a^2\{(\cos 2\pi t_1 - \cos 2\pi t_2)^2 + (\sin 2\pi t_1 - \sin 2\pi t_2)^2\}) \\ &= \exp\{-4a^2 \sin^2(\pi t_1 - \pi t_2)\} \end{aligned}$$

is the so-called *squared exponential periodic kernel* obtained by mapping the squared exponential kernel on unit interval $[0, 1]$ to the circle through $Q : [0, 1] \to \mathbb{S}$, $\omega \to (\cos 2\pi\omega, \sin 2\pi\omega)$ as in MacKay (1998). The parameters a and τ control the smoothness and scale of the kernel, respectively. As shown in Li and Ghosal (2015), the covariance kernel has the following closed form decomposition: $G_a(t, t') = \sum_{k=1}^{\infty} v_k(a)\psi_k(t)\psi_k(t')$ where

$$v_1(a) = e^{-2a^2} I_0(2a^2); \quad v_{2j}(a) = v_{2j+1}(a) = e^{-2a^2} I_j(2a^2),\ j \geq 1;$$

and $I_n(x)$ denotes the modified Bessel function of the first kind of order n and $\psi_j(t)$ is the jth Fourier basis function in $\{1, \cos 2\pi t, \sin 2\pi t, ...\}$. The above expansion allows us to write the boundary as a sum of basis functions:

$$\gamma(\omega) = \mu(\omega) + \sum_{k=1}^{\infty} z_k \psi_k(\omega), \tag{2}$$

where $z_k \sim \mathsf{N}(0, v_k(a)/\tau)$. In practice, we truncate this basis function expansion using the first L functions, i.e., $\gamma(\omega) = \mu(\omega) + \sum_{k=1}^{L} z_k \psi_k(\omega)$. In the **BayesBD** package, we use $L = 2J + 1$ with the default $J = 10$, which seems adequate for accurate approximation of $\gamma(w)$ as shown in Li and Ghosal (2015), but users may specify a different value depending on the application.

We use a Gamma prior distribution Gamma(α_a, β_a) for the rescaling factor a. This random rescaling scheme is critical to obtain rate adaptive estimates without assuming the smoothness level α in Assumption 3 is known; see, for example, van der Vaart and van Zanten (2009) and Li and Ghosal (2015). The default values of hyperparameters are $\alpha_a = 2$ and $\beta_a = 1$.

We use a constant function as the prior mean function $\mu(\cdot)$, with value determined by user input or by an initial maximum likelihood estimation. The other hyperparameter and parameters follow standard conjugate priors. Specifically, we use a Gamma distribution Gamma$(\alpha_\tau, \beta_\tau)$ prior for τ with default values $\alpha_\tau = 500$ and $\beta_\tau = 1$. Priors for the nuisance parameters ξ and ρ depend on the parametric family f, which are:

- For binary images: the parameters are the probabilities $(\pi_1, \pi_2) \sim \text{OIB}(\alpha_1, \beta_1, \alpha_1, \beta_1)$, where OIB stands for ordered independent Beta distributions.
- For Gaussian noise: the parameters are the mean and standard deviation $(\mu_1, \sigma_1, \mu_2, \sigma_2)$ with prior distributions $(\mu_1, \mu_2) \sim \text{OIN}(\mu_0, \sigma_0^2, \mu_0, \sigma_0^2)$ and $(\sigma_1^{-2}, \sigma_2^{-2}) \sim \text{OIG}(\alpha_2, \beta_2, \alpha_2, \beta_2)$, where OIN and OIG are ordered independent normal and Gamma distributions, respectively.

The orders in OIB, OIN and OIG are provided by users if such information is available; otherwise, the ordered independent distributions revert to independent distributions. Our default specifications are chosen to make the corresponding prior distributions spread out. For example, in the **BayesBD** package, the default values are $\alpha_1 = \beta_1 = 0$ for binary images, and $\mu_0 = \bar{Y}, \sigma_0 = 10^3$ and $\alpha_2 = \beta_2 = 10^{-2}$ for Gaussian noise, where $\bar{Y}$ is the same mean of all intensities. Under Assumptions 1–4, Li and Ghosal (2015) proved that the nonparametric Bayes approach is (nearly) rate-optimal in the minimax sense, adaptive to unknown smoothness level α.

Posterior sampling and estimation of the boundary

Let $z = \{z_i\}_{i=1}^L$ and $\Sigma_a = \text{diag}(v_1(a), \ldots, v_L(a))$. We use Metropolis-Hastings (MH) with the Gibbs sampler (Geman and Geman, 1984) to sample the joint distribution of the parameters (z, ξ, ρ, τ, a), where the MH step is for the vector parameter z. We also allow a slice sampling with the Gibbs sampler where slice sampling is used for z as in Li and Ghosal (2015). We give the detailed sampling algorithms for binary image in Algorithm 1 and Gaussian-noised images in Algorithm 2. Comparisons between MH and slice sampling, along with other numerical performances are referred to in the section on **Performance tests**.

Algorithm 1 – Binary images.

Initialize the parameters: $z = 0$, $\tau = 500$, $a = 1$, and $\mu(\cdot)$ is taken to be constant, i.e. a circle. Then, initialize $(\xi, \rho) = (\pi_1, \pi_2)$ by the maximum likelihood estimates given $\mu(\cdot)$.

1. At iteration $t+1$, sample $z^{(t+1)}|(\pi_1^{(t)}, \pi_2^{(t)}, \tau^{(t)}, a^{(t)}, Y, X)$ one entry at a time, using either MH sampling and slice sampling, using the following logarithm of the conditional posterior density

$$N_1 \log \frac{\pi_1^{(t)}(1-\pi_2^{(t)})}{\pi_2^{(t)}(1-\pi_1^{(t)})} + n_1 \log \frac{1-\pi_1^{(t)}}{1-\pi_2^{(t)}} - \frac{\tau}{2}(z^{(t)})^\top \Sigma_{a^{(t)}}^{-1} z^{(t)},$$

   where $n_1 = \sum_{i=1}^n 1(r_i < \gamma_i^{(t)})$ and $N_1 = \sum_{i=1}^n 1(r_i < \gamma_i^{(t)}) Y_i$; here (r_i, ω_i) are the polar coordinates of pixel location X_i and $\gamma_i^{(t)} = \gamma^{(t)}(\omega_i)$ is the radius of the image boundary at iteration t and the ith pixel.

2. Sample $\tau^{(t+1)}|z^{(t+1)}, a^{(t)} \sim \text{Gamma}(\alpha^\star, \beta^\star)$ where $\alpha^\star = \alpha_\tau + L/2$ and $\beta^\star = \beta_\tau + (z^{(t+1)})^\top \Sigma_{a^{(t)}}^{-1} z^{(t+1)}/2$.

3. Sample $(\pi_1, \pi_2)|(z, Y) \sim \text{OIB}(\alpha_1 + N_1, \beta_1 + n_1 - N_1, \alpha_1 + N_2, \beta_1 + n_2 - N_2)$, where $n_2 = \sum_{i=1}^n 1(r_i \geq \gamma_i^{(t)})$ and $N_2 = \sum_{i=1}^n 1(r_i \geq \gamma_i^{(t)}) Y_i$.

4. Sample $a^{(t+1)}|(z^{(t+1)}, \tau^{(t+1)})$ by slice sampling using the logarithm of the conditional posterior density

$$-\sum_{k=1}^L \frac{\log v_k(a^{(t)})}{2} - \sum_{k=1}^L \frac{\tau^{(t+1)} z_k^2}{2 v_k(a^{(t)})} + (\alpha_a - 1) \log a^{(t)} - \beta_a.$$

Let $\{\gamma_t(\omega)\}_{t=1}^T$ be the posterior samples after burn-in where T is the number of posterior samples. We use the posterior mean as the estimate and construct a variable-width uniform credible band. Specifically, let $(\widehat{\gamma}(\omega), \widehat{s}(\omega))$ be the posterior mean and standard deviation functions derived from $\{\gamma_t(\omega)\}$. For the tth MCMC run, we calculate the distance $u_t = \|(\gamma_t - \widehat{\gamma})/s\|_\infty = \sup_\omega \{|\gamma_t(\omega) - \widehat{\gamma}(\omega)|/\widehat{s}(\omega)\}$ and obtain the 95th percentile of all the u_t's, denoted as L_0. Then a 95% uniform credible band is given by $[\widehat{\gamma}(\omega) - L_0\widehat{s}(\omega), \widehat{\gamma}(\omega) + L_0\widehat{s}(\omega)]$.

Algorithm 2 – Gaussian-noised images.

Initialize the parameters: $z = 0$, $\tau = 500$, $a = 1$, and $\mu(\cdot)$ is taken to be constant, i.e. a circle. Then, initialize $(\xi, \rho) = (\pi_1, \pi_2)$ by the maximum likelihood estimates given $\mu(\cdot)$.

1. At iteration $t+1$, sample $z^{(t+1)}|(\mu_1^{(t)}, \sigma_1^{(t)}, \mu_2^{(t)}, \sigma_2^{(t)}, \tau^{(t)}, a^{(t)}, Y, X)$ one entry at a time, using either slice sampling or MH sampling, using the following logarithm of the conditional posterior density

$$-n_1(\log\sigma_1^{(t)} - \log\sigma_2^{(t)}) - \sum_{i\in I_1}\frac{(Y_i-\mu_1^{(t)})^2}{2(\sigma_1^2)^{(t)}} - \sum_{i\in I_2}\frac{(Y_i-\mu_2^{(t)})^2}{2(\sigma_2^2)^{(t)}} - \frac{\tau(z^{(t)})^\top\Sigma_{a^{(t)}}^{-1}z^{(t)}}{2}.$$

2. Sample $\tau^{(t+1)}|z^{(t+1)}, a^{(t)}$ as in Algorithm 1.

3. Sample $(\mu_1, \sigma_1, \mu_2, \sigma_2)|(z, Y)$ conjugately.

   - Sample $(\sigma_1^{-2})^{(t+1)}$ from a Gamma distribution with parameters

$$\alpha = \alpha_2 + \frac{n_1}{2}, \qquad \beta = \beta_2 + \sum_{i\in I_1}\frac{(Y_i-\bar{Y}^{(1)})^2}{2} + \frac{\sigma_0^{-2}n_1}{n_1+\sigma_0^{-2}}\frac{(\bar{Y}^{(1)}-\mu_0)^2}{2},$$

     where $\bar{Y}^{(1)}$ is the sample mean of intensities in I_1.

   - sample $\mu_1^{(t+1)}$ from a normal distribution with mean and standard deviation

$$\frac{\sigma_0^{-2}\mu_0}{n_1+\sigma_0^{-2}} + \frac{n_1\bar{y}_1}{n_1+\sigma_0^{-2}}, \qquad (n_1+\sigma_0^{-2})^{-1/2}.$$

   - Sample $(\sigma_2^{-2})^{(t+1)}$ and $\mu_2^{(t+1)}$ analogously.
   - If `ordering` information is available, sort $(\mu_1^{(t+1)}, \mu_2^{(t+1)})$ and $(\sigma_1^{(t+1)}, \sigma_2^{(t+1)})$ accordingly.

4. Sample $a^{(t+1)}|(z^{(t+1)}, \tau^{(t+1)})$ as in Algorithm 1.

Analysis of image boundaries using BayesBD from the command line

There are three steps to our Bayesian image boundary analysis: load the image data into R in the appropriate format, use the functions provided to sample from the joint posterior of the full parameter $\theta = (\xi, \rho, \gamma)$, and summarize the posterior samples both numerically and graphically.

Generating image data

Two functions are included in **BayesBD** to facilitate data simulation for numerical experiments: `par2obs` for binary images and `parnormobs` for Gaussian-noised images. Table 1 describes the function arguments to `par2obs`, which returns sampled intensities and pixel locations in both Euclidean and polar coordinates. The function `parnormobs` is similar, with the replacement of arguments `pi.in` and `pi.out` by `mu.in`, `mu.out`, `sd.in`, and `sd.out` corresponding to parameters μ_1, μ_2, σ_1, and σ_2, respectively.

As a demonstration, the following code generates a 100×100 binary image of an ellipse using a jitteredly-random design with a reference point of (0.5, 0.5):

```
> gamma.fun = ellipse(a = 0.35, b = 0.25)
> bin.obs = par2obs(m = 100, pi.in = 0.6, pi.out = 0.4, design = 'J',
+ gamma.fun, center = c(0.5, 0.5))
```

Similarly, the following code generates a 100×100 Gaussian-noised image with a triangle boundary using a random uniform design with a reference point of (0.5, 0.5):

```
> gamma.fun = triangle2(0.5)
```

```
> norm.obs = parnormobs(m = 100, mu.in = 1, mu.out = -1, sd.in = 1,
+ sd.out = 1, design = 'U', gamma.fun, center = c(0.5, 0.5))
```

The output of either `par2obs` or `parnormobs` is a list containing the intensities Y in a vector named `intensity`, the vectors `theta.obs` and `r.obs` containing the polar coordinates of the pixel locations X, and the reference point contained in a vector named `center`. Image data to be analyzed using **BayesBD** can be in a list of this form, or may be a `.png` or `.jpg` image file.

Argument	Description
`m`	Generate $m \times m$ observations over the unit square.
`pi.in`	The success probability, $P(Y_i = 1)$, if $X_i \in I_1$.
`pi.out`	The success probability, $P(Y_i = 1)$, if $X_i \in I_2$.
`design`	Determines how locations X_i are determined: 'D' for deterministic (equally-spaced grid) design, 'U' for completely uniformly random, or 'J' for jitteredly random design.
`center`	Two-dimensional vector of Euclidean coordinates (x,y) of the reference point in I_1.
`gamma.fun`	A function to generate boundaries, e.g. ellipse or triangle2.

Table 1: Arguments of the `par2obs` function.

Analysis and visualization

There are two functions to draw posterior samples following Algorithm 1 and 2 based on images either simulated or provided by users: `fitBinImage` for binary images, and `fitContImage` for Gaussian-noised images. These sampling functions take the same arguments, with the exception of the `ordering` input which is duplicated in `fitContImage` to allow ordering of the two parameters, i.e., the mean and standard deviation. The inputs for `fitBinImage` are summarized in Table 2. We have included a function `rectToPolar` to facilitate formatting the image data for `fitBinImage` and `fitContImage` by converting the rectangular coordinates of the pixels to polar coordinates. The initial boundary is a circle with radius `inimean` and center `center`. The radius `inimean` may be specified by the user or left blank, in which case it will be estimated using maximum likelihood.

`image`	The noisy observation, either a list with elements: `intensity`, a vector of intensities; `theta.obs` a vector of pixel radian measure from `center`; `r.obs` a vector of pixel radius measure from `center`; or a string giving the path to a `.png` or `.jpg` file.
`gamma.fun`	the true boundary, if known, used for plotting.
`center`	the reference point in Euclidean coordinates.
`inimean`	A constant specifying the initial boundary μ. Defaults to NULL, in which case μ is estimated automatically using maximum likelihood.
`nrun`	The number of MCMC runs to keep for estimation.
`nburn`	The number of initial MCMC runs to discard.
`J`	The number of eigenfunctions to use in estimation is $2J + 1$.
`ordering`	Indicates which Bernoulli distribution has higher success probability: `"I"`, the Bernoulli distribution inside the boundary; `"O"`, ther Bernoulli distribution outside the boundary; `"N"`, no ordering information is available.
`mask`	Logical vector (same length as `obs$intensity`) to indicate region of interest. Should this data point be included in the analysis? Defaults to NULL and uses all data points.
`slice`	Should slice sampling be used to sample Fourier basis function coefficients?
`outputAll`	Should all posterior samples be returned?

Table 2: Arguments of the `fitBinImage` function.

If argument `outputAll` is `FALSE`, the functions `fitBinImage` and `fitContImage` produce output vectors `theta`, `estimate`, `upper`, and `lower` giving a grid of 200 values on $[0, 2\pi]$ on which the boundary

γ is estimated, along with 95% uniform credible bands. If argument `outputAll` is `TRUE`, the functions also return posterior samples of (ξ, ρ) and Fourier basis function coefficients z.

Following the examples of data simulation for binary and Gaussian-noised images in the previous section, we can obtain posterior samples via

```
> bin.samples= fitBinImage(bin.obs, c(.5, .5), NULL, 4000, 1000, 10, "I", NULL, TRUE,
+ FALSE)
```

for a binary image and

```
> norm.samples = fitContImage(norm.obs, c(.5, .5), NULL, 4000, 1000, 10, "I", "N", NULL,
TRUE, FALSE)
```

for a Gaussian-noised image. For each sampling function, we have set the center of the image as $(0.5, 0.5)$, instructed the sampler to use a mean function of $\mu(\cdot) = 0.4$, keep 4000 samples, burn 1000 samples, use $L = 2 \times 10 + 1 = 21$ basis functions to model γ, use slice sampling for the basis function coefficients, and return only the plotting results.

Using the function `plotBD` we can easily construct plots of our model results. There are two arguments to `plotBD`: `fitted.image` is a list of results from `fitBinImage` or `fitContImage`, and `plot.type` which indicates whether to plot the image data only, the posterior mean and credible bands, or the posterior mean overlaid on the image data. Using the posterior samples we obtained from the Gaussian-noised image above, we can plot our results with the following code

```
> par(mfrow = c(1, 2))
> plotBD(norm.samples, 1)
> plotBD(norm.samples, 2)
```

to produce Figure 2.

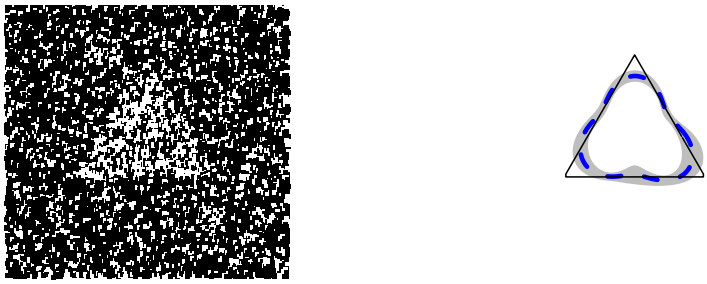


Figure 2: Output from `fitBinImage` as plotted using `plotBD`. The plot on the left is the simulated binary observation, and the plot on the right is the estimated boundary and 95% uniform credible bands.

BayesBD shiny app

In order to reach a broad audience of users, including those who may not be familiar with R, we have created a **shiny** app version of **BayesBD** to implement the boundary detection method. The app has the full functionality to reproduce the simulations in Section 2.5 and conduct real data applications using images uploaded by users. The app is accessible both from the package by running the code `BayesBDshiny()` in an R session or externally by visiting `https://syring.shinyapps.io/shiny/`. With the app users can analyze real data or produce a variety of simulations.

Once the app is open, users are presented with an array of inputs to set as illustrated in Figure 3. In order to analyze real data, the user should select "user continuous image" or "user binary image" from the second drop-down menu. Next, the user inputs the system path to the `.png` or `.jpg` file. A plot of the image will appear and the user will be prompted to identify the image center with a mouse click. The user has some control over the posterior sample size, but we recommend to first limit the number of available samples in order to display results quickly. Finally, the user may enter ordering information for the mean and variance of pixel intensities at the bottom of the display.

For simulations, we first select either an elliptical or triangular boundary, or upload an R script with a custom boundary function. Next, the user instructs the app to either simulate a binary or Gaussian-noised image, or to use binary or Gaussian data the user has uploaded. In addition to the

boundary function, the user specifies the reference point. Sample sizes for simulations are kept at 100×100 pixels. Finally, the last several inputs allow the user to customize the intensity parameters (ξ, ρ) for binary and Gaussian simulations. Once the user has selected all settings, clicking the "Update" button at the bottom of the window will run the posterior sampling algorithm, which should take less than a minute on a typical computer for image data of 100×100 pixels. The "Download" button provides the user with a file indicating which pixels were contained inside the estimated boundary as determined by the outer edge of the 95% uniform credible bands.

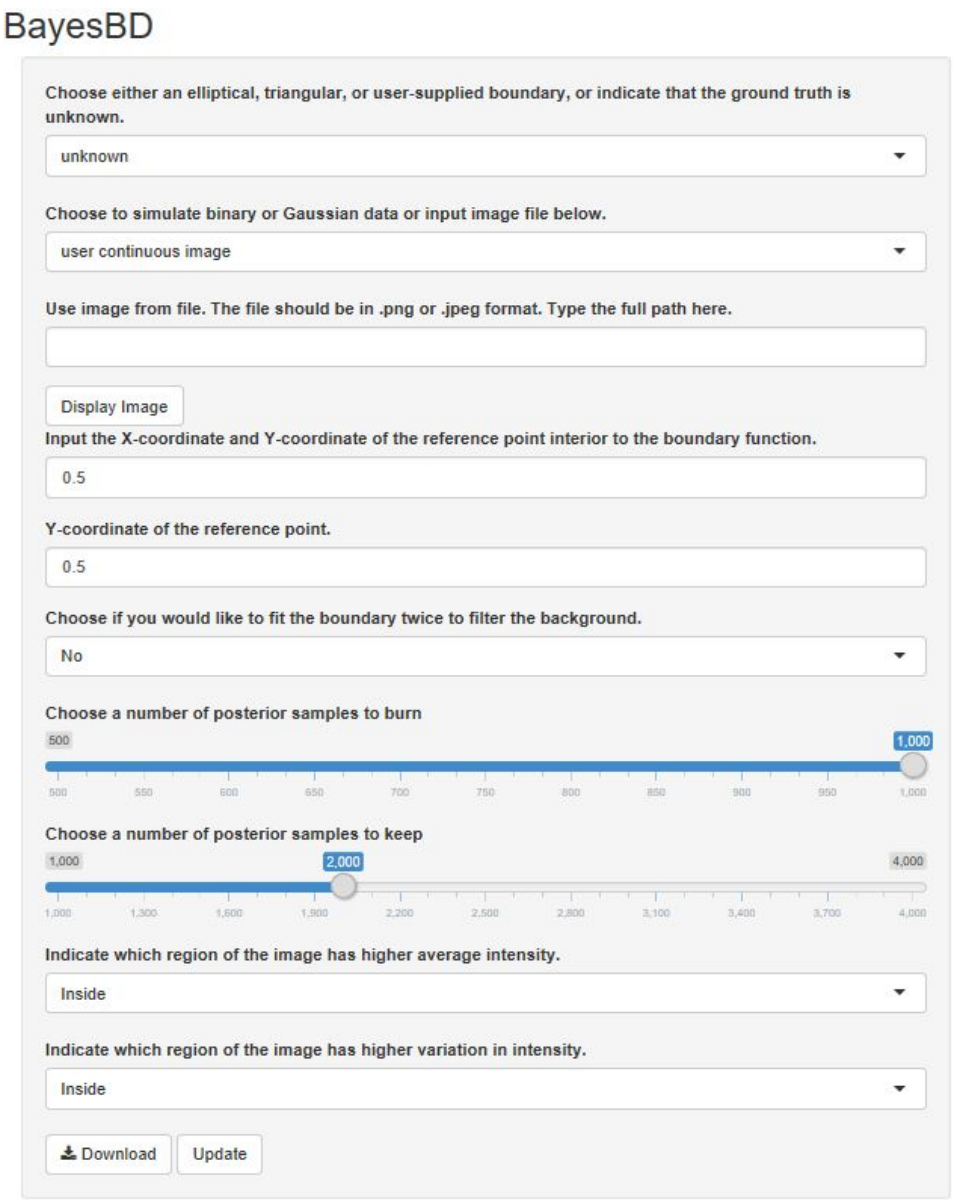


Figure 3: Screenshot of shiny app implementing **BayesBD**.

Performance tests

Comparison of sampling methods

The main aim of this section is to highlight the speed improvements we have made in the latest version of **BayesBD**. Our flexibility in choosing between slice and Metropolis-Hastings (MH) sampling algorithms gives users the potential to unlock efficiency gains. We highlight these gains below for both binary and Gaussian-noised simulations, and note that the faster MH method suffers little in accuracy.

In our performance tests, we consider the following examples, which correspond to examples B2, B3, and G1 from Li and Ghosal (2015).

S1. Image is an ellipse centered at $(0.1, 0.1)$ and rotated 60° counterclockwise. Intensities are binary with $\pi_1 = 0.5$ and $\pi_2 = 0.2$.

S2. Image is an equilateral triangle centered at $(0, 0)$ with height 0.5. Intensities are binary with $\pi_1 = 0.5$ and $\pi_2 = 0.2$.

S3. Image is an ellipse centered at $(0.1, 0.1)$ and rotated 60° counterclockwise. Intensities are Gaussian with $\mu_1 = 4$, $\sigma_1 = 1.5$, $\mu_2 = 1$, and $\sigma_2 = 1$.

We simulated each case 100 times using $n = 100 \times 100$ observations per simulation. In each run, we sampled 4000 times from the posterior after a 1000 sample burn in. The results of our performance tests comparing slice with MH sampling are summarized in Table 3.

If Metropolis-Hastings sampling is used instead of slice sampling, we observe a speed up by about a factor of two for binary images, seven for the Gaussian-noised image. Slice sampling is guaranteed to produce unique posterior samples, and may give better results than Metropolis-Hastings samplers, especially when Metropolis-Hastings mixes poorly and produces many repeated samples. However, slice sampling may involve a very large number of proposed samples for each accepted sample, requiring many likelihood evaluations. On the other hand, each Metropolis Hastings sample requires only two evaluations of the likelihood.

To measure the accuracy of **BayesBD** we use three metrics: the Lebesgue error, which is simply the area of the symmetric difference between the posterior mean boundary and the true boundary; the Dice Similarity Coefficient(DSC), see Feng and Tierney (2015); and Hausdorff distance, see Thompson (2014) and Thompson (2017). We have included the utility functions `lebesgueError`, `dsmError`, and `hausdorffError`, which take as input the output of either `fitBinImage` or `fitContImage` and output the corresponding error. In the binary image examples considered, the different sampling algorithms did not affect the accuracy of the posterior mean boundary estimates when measured by Lebesgue error, i.e., the area of the symmetric difference between the estimated boundary and the true boundary. For the Gaussian-noised image, the slice sampling method produced Lebesgue errors approximately an order of magnitude smaller than when using Metropolis-Hastings sampling, but the overall size of the errors was still small in both cases and practically indistinguishable when plotting results. With our built-in functions, it is easy to reproduce Example S2 in Table 3 with the following code:

```
> gamma.fun = triangle2(0.5)
> for(i in 1:100){
+         norm.obs = par2obs(m = 100, pi.in = 0.5, pi.out = 0.2, design = 'J',
+                                          center = c(0.5, 0.5), gamma.fun)
+         norm.samp.MH = fitBinImage(norm.obs, gamma.fun, NULL, NULL, 4000, 1000,
+                                          10, "I", rep(1, 10000), FALSE, FALSE)
+         norm.samp.slice = fitBinImage(norm.obs, gamma.fun, NULL, NULL, 4000,
+                                           1000, 10, "I", rep(1, 10000), TRUE, FALSE)
+         print(c(dsmError(norm.samp.MH), hausdorffError(norm.samp.MH),
+                 lebesgueError(norm.samp.MH), dsmError(norm.samp.slice),
+                 hausdorffError(norm.samp.slice), lebesgueError(norm.samp.slice)))
+ }
```

Example	Sampling Method	Runtime (s)	Lebesgue Error	DSC	Hausdorff
S1	MH	58	0.01(0.01)	0.02(0.00)	0.02(0.00)
	Slice	100	0.01(0.00)	0.02(0.00)	0.02(0.00)
S2	MH	45	0.02(0.00)	0.09(0.02)	0.09(0.01)
	Slice	82	0.02(0.00)	0.09(0.01)	0.09(0.01)
S3	MH	66	0.01(0.01)	0.01(0.01)	0.01(0.01)
	Slice	488	0.00(0.00)	0.01(0.01)	0.01(0.01)

Table 3: Average Runtimes (in seconds) and Lebesgue errors (with standard deviations) of posterior mean boundary estimates using C++.

Comparison of coding platforms

Our use of C++ for posterior sampling has led to very significant efficiency gains over using R alone. Implementation of C++ with R is streamlined using **Rcpp** and **RcppArmadillo**.

The first implementation of **BayesBD** (version 0.1 in Li (2015)) was entirely written in R. The results in Table 4 labeled R code reflect this first version of the package, while those labeled C++ code

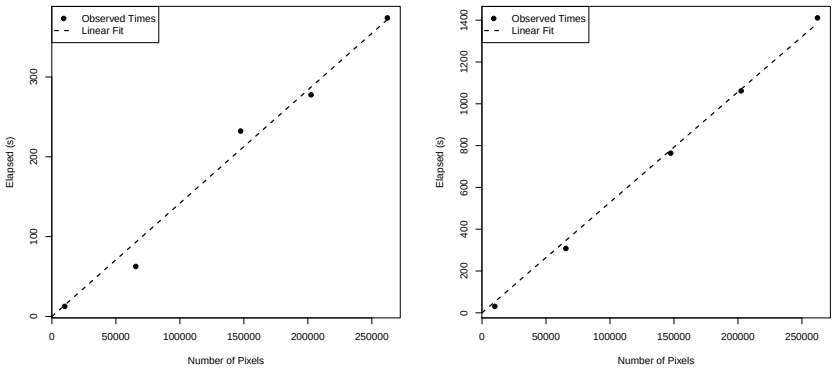


Figure 4: Left: Runtimes of `fitBinImage`. Right: Runtimes of `fitContImage`

are from the new version. The main takeaway is that the new code developed using C++ is at least three times faster in the binary image examples and over six times faster for the Gaussian-noised image example, both measured while using slice sampling.

Example	Coding Method	Runtime (s)
S1	C++	100
	R	375
S2	C++	82
	R	246
S3	C++	488
	R	3327

Table 4: Average run times (in seconds) using slice sampling.

Scalability of BayesBD

The Gaussian process is notorious for scaling poorly as n increases because it is usually necessary to invert of a large covariance matrix. By utilizing the analytical decomposition of a GP kernel in (2), we eliminates the step of inverting a n by n covariance matrix and the **BayesBD** package appears to achieve a linear complexity. We investigate the scalability of **BayesBD** by plotting the system time against sample size for `fitBinImage` and `fitContImage` in Figure 4 using a triangular boundary curve and 5000 MCMC iterations. Both algorithms appear to scale approximately linearly in number of pixels, which makes sense as the costliest computations in Steps 1 and 3 in Algorithms 1 and 2 only involve sums over the n pixels.

Comparison with existing packages

Although no packages besides **BayesBD** provide boundary estimation, there are several existing packages that can provide image segmentation or filtering. Below we make some qualitative comparisons between **BayesBD** and **mritc**, **bayesImageS**, and **bayess**; see Feng and Tierney (2015), Moores et al. (2017), and Robert and Marin (2015) in a later section. Figure 5 compares these packages using two simulated images with Bernoulli and Gaussian noise, respectively. **BayesBD** gives very reasonable estimates for the true boundaries; **mritc** package fails to deliver a recognizable smoothed image in either example; and **bayesImageS** was able to produce a very clear segmentation for the Gaussian-noised ellipse example, but not for the triangle image with Bernoulli noise.

Real data application

Medical imaging

Chen et al. (2006) studied the performance of two different tracers used in conjunction with positron emission tomography (PET) scans in identifying brain tumors. Figure 6, reproduced from (Chen

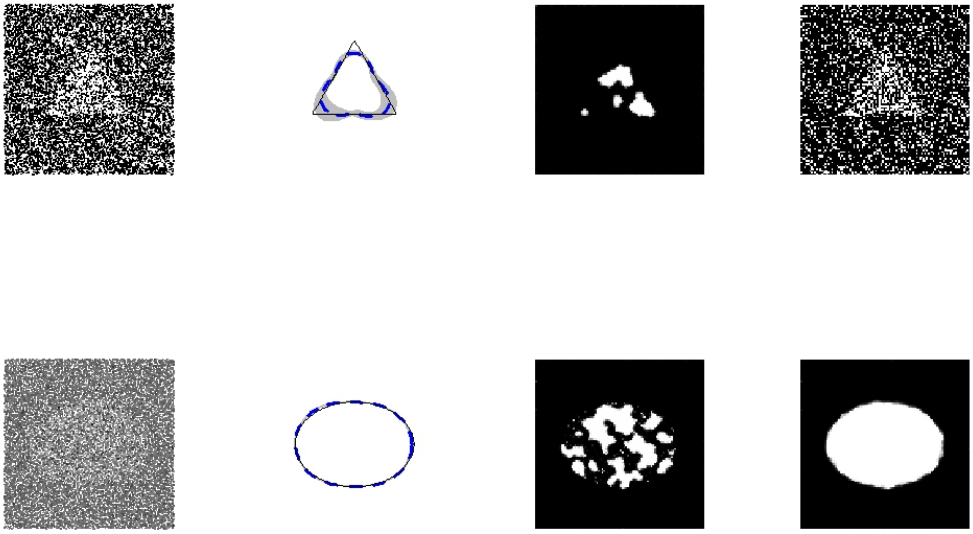


Figure 5: Left to right: the image, the **BayesBD** boundary estimate, the **mritc** filtered image, and the **bayesImageS** filtered image.

et al., 2006), gives an example of the image data used in diagnosing tumors, and demonstrates their conclusion that the F-FDOPA tracer provides a more informative PET scan than the F-FDA tracer. We use the **BayesBD** package to analyze the F-FDOPA PET scan images in Figure 6. The tumor imaging data along with sample code for reproducing the following analysis can be found in the documentation to the **BayesBD** package.

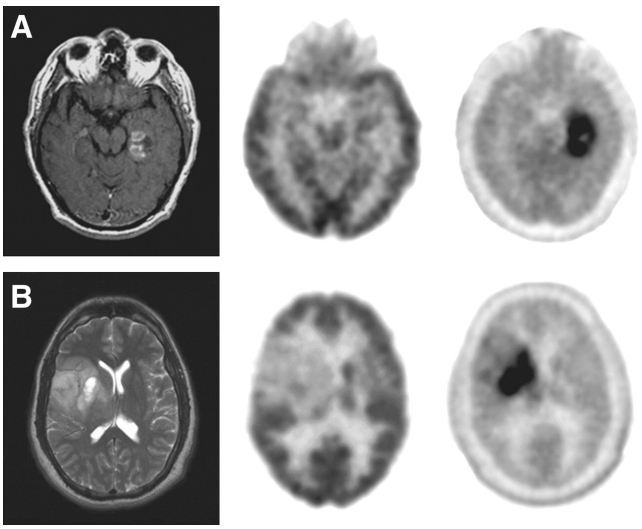
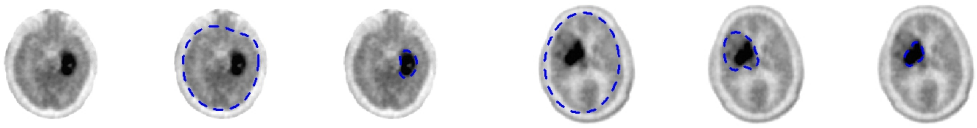


Figure 6: MRI (left), F-FDG PET (middle), and F-FDOPA PET (right) of glioblastoma (A) and grade II oligodendroglioma (B). Image taken from Chen et al. (2006).

We convert the two F-FDOPA PET images in Figure 6 into 111×111-pixel images and normalize the intensities to the interval [0,10]. The pixel coordinates are a grid on $[0,1] \times [0,1]$ and we choose reference points $(0.7, 0.5)$ and $(0.4, 0.55)$ for each image, roughly corresponding to the center of the darkest part of each image. We use the default mean function, choose $J = 10$ for 21 basis functions, and sample 4000 times after a 1000 burn-in using MH sampling.

Figure 7 displays posterior mean boundary estimates for the F-FDOPA images in Figure 6. From the analysis on glioblastoma (A) in the first two plots, it seems that that we accurately capture the regions



A. Glioblastoma B. Grade II oligodendroglioma

Figure 7: F-FDOPA PET images from Chen et al. (2006) (left) fit twice, and (right) fit three times to filter the background and find features at increasing granularity.

of interest in the F-FDOPA PET images. Furthermore, it is expected that the Gaussian assumption on the real data may fail, and this shows that the method implemented in **BayesBD** is robust to model misspecifications, thus practically useful.

Tumor heterogeneity, which is not unusual in many applications, may make the boundary detection problem more challenging (Heppner, 1984; Ananda and Thomas, 2012). The **BayesBD** package allows us to address tumor heterogeneity by a repeated implementation. We first apply `fitContImage` to the entire image, which includes a white background not of interest, and produce the estimated boundary. This step succeeds in separating the brain scan from the white background. A second run is performed on the subset of the image inside the outer 95% uniform credible band, producing a nested boundary. In general, this technique can be used in a multiple region setting where the data displays more heterogeneity than the simple "image and background" setup in (1).

Satellite imaging

We compared the performance of **BayesBD** with the R packages **mritc** and **bayess** using an image of Lake Menteith available in the **bayess** package. **BayesBD** gives a very reasonable estimate for the boundary of the lake even though it is not smooth. The **mritc** package again does not provide useful output in this example, but **bayess** produces a nicely-segmented image; see Figure 8.

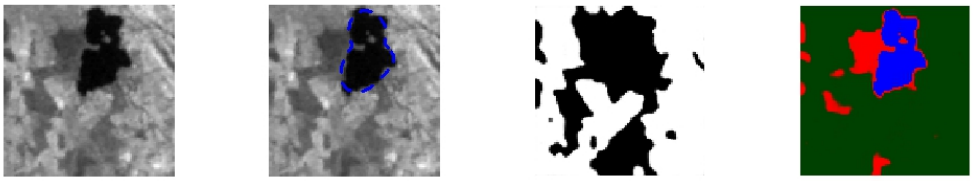


Figure 8: Left to right: the image, the **BayesBD** boundary estimate, the **mritc** filtered image, and the **bayess** segmented image.

Summary

BayesBD is a new computational platform for image boundary analysis with many advantages over existing software. The underlying methods in functions `fitBinImage` and `fitContImage` are based on theoretical results ensuring their dependability over a range of problems. Our use of **Rcpp** and **RcppArmadillo** help make **BayesBD** much faster than base R code and further speed can be gained by our flexible sampling algorithms. Finally, our integration with **shiny** provides users with an easy way to utilize our package without having to write R code.

For the latest updates to **BayesBD** and requests, readers are recommended to check out the package page at CRAN or refer to the Github page at `https://github.com/nasyring/GSOC-BayesBD`.

Acknowledgments

This work was partially supported by the Google Summer of Code program. We thank the Associate Editor and one anonymous referee for comprehensive and constructive comments that helped to improve the paper.

Bibliography

R. S. Ananda and T. Thomas. Automatic segmentation framework for primary tumors from brain MRIs using morphological filtering techniques. In *Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on*, pages 238–242. IEEE, 2012. [p160]

M. Basu. Gaussian-based edge-detection methods—a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 32(3):252–260, 2002. [p149]

S. Bhardwaj and A. Mittal. A survey on various edge detector techniques. *Procedia Technology*, 4: 220–226, 2012. [p149]

W. Chen, D. H. S. Silverman, D. Sibylle, J. Czernin, N. Kamdar, W. Pope, N. Satyamurthy, C. Schiepers, and T. Cloughesy. F-FDOPA PET imaging of brain tumors: Comparison study with F-FDG PET evaluation of diagnostic accuracy. *Journal of Nuclear Medicine*, 47(6):904–911, 2006. [p158, 159, 160]

L. Cucala and J.-M. Marin. "bayesian inference on a mixture model with spatial dependence. *J. Comput. Graph. Stat.*, 22(3):584–597, 2014. [p149]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer-Verlag, New York, 2013. ISBN 978-1-4614-6867-7. [p149]

D. Eddelbuettel and C. Sanderson. Rcpparmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics & Data Analysis*, 71:1054–1063, 2014. URL http://dx.doi.org/10.1016/j.csda.2013.02.005. [p149]

D. Feng and L. Tierney. *Mritc: MRI Tissue Classification*, 2015. URL https://CRAN.R-project.org/package=mritc. [p157, 158]

M. C. Fitzpatrick, E. L. Preisser, A. Porter, J. Elkinton, L. A. Waller, B. P. Carlin, and A. M. Ellison. Ecological boundary detection using Bayesian areal wombling. *Ecology*, 91(12):3448–3455, 2010. ISSN 00129658. URL http://www.jstor.org/stable/29779525. [p149]

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984. [p152]

U. Grenander and M. I. Miller. *Pattern Theory: From Representation to Inference.* Oxford University Press, New York, 2007. [p149]

W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. [p149]

G. H. Heppner. Tumor heterogeneity. *Cancer research*, 44:2259–2265, 1984. [p160]

M. A. Hurn, O. K. Husby, and H. Rue. Advances in Bayesian image analysis. In P. J. Green, N. L. Hjort, and S. Richardson, editors, *Highly Structured Stochastic Systems*, pages 301–322. Oxford University Press, Oxford, 2003. [p149]

C. Li, C. Xu, C. Gui, and M. D. Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE Trans. Image Processing*, 19(12):3243–3254, 2010. [p149]

M. Li. *BayesBD: Bayesian Boundary Detection in Images*, 2015. R package version 0.1. [p149, 157]

M. Li and S. Ghosal. Bayesian detection of image boundaries, 2015. URL https://arxiv.org/abs/1508.05847. [p149, 150, 151, 152, 156]

H. Lu and B. P. Carlin. Bayesian areal wombling for geographical boundary analysis. *Geographical Analysis*, 37(3):265–285, 2005. URL http://proxying.lib.ncsu.edu/index.php?url=/docview/289033244?accountid=12725. [p149]

D. J. MacKay. Introduction to Gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998. [p151]

R. Maini and H. Aggarwal. Study and comparison of various image edge detection techniques. *International Journal of Image Processing*, 3(1):1–11, 2009. [p149]

J.-M. Marin and C. P. Robert. *Bayesian Essentials with R*. Springer-Verlag, 2014. [p149]

M. Moores, K. Mengersen, and D. Feng. *Bayesian Methods for Image Segmentation Using a Potts Model*, 2017. URL https://CRAN.R-project.org/package=bayesImageS. [p158]

R. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003. [p149]

C. P. Robert and J.-M. Marin. *Bayesian Essentials with R*, 2015. URL https://CRAN.R-project.org/package=bayess. [p158]

RStudio, Inc. *Easy Web Applications in R.*, 2016. URL: http://www.rstudio.com/shiny/. [p149]

N. Syring and M. Li. *BayesBD: Bayesian Inference for Image Boundaries*, 2017. R package version 1.2. [p149]

R. F. Thompson. RadOnc: An R package for analysis of dose-volume histogram and three- dimensional structural data. *J. Rad. Onc. Info.*, 6(1):98–100, 2014. [p157]

R. F. Thompson. *RadOnc: Analytical Tools for Radiation Oncology*, 2017. URL https://CRAN.R-project.org/package=RadOnc. [p157]

A. W. van der Vaart and J. H. van Zanten. Adaptive Bayesian estimation using a Gaussian random field with inverse gamma bandwidth. *Ann. Statist.*, 37(5B):2655–2675, 2009. ISSN 0090-5364. URL https://doi.org/10.1214/08-aos678. [p151]

L. A. Waller and C. A. Gotway. *Applied Spatial Statistics for Public Health Data*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2004. ISBN 0-471-38771-1. URL https://doi.org/10.1002/0471662682. [p149]

W. Yuan, K. S. Chin, M. Hua, G. Dong, and C. Wang. Shape classification of wear particles by image boundary analysis using machine learning algorithms. *Mechanical Systems and Signal Processing*, 72-73:346–358, 2016. [p149]

D. Ziou and S. Tabbone. Edge detection techniques – an overview. *Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii*, 8:537–559, 1998. [p149]

Nicholas Syring
Department of Statistics
North Carolina State University, 5109 SAS Hall
2311 Stinson Dr.
Raleigh, NC 27695 USA
nasyring@ncsu.edu

Meng Li
Rice University
Department of Statistics
6100 Main St
Houston, TX 77251-1892 USA
meng@rice.edu