Leave Pima Indians Alone: Binary Regression as a Benchmark for Bayesian Computation

Nicolas Chopin, James Ridgway

LEAVE PIMA INDIANS ALONE: BINARY REGRESSION AS A BENCHMARK FOR BAYESIAN COMPUTATION

NICOLAS CHOPIN AND JAMES RIDGWAY

Abstract. Whenever a new approach to perform Bayesian computation is introduced, a common practice is to showcase this approach on a binary regression model and datasets of moderate size. This paper discusses to which extent this practice is sound. It also reviews the current state of the art of Bayesian computation, using binary regression as a running example. Both sampling-based algorithms (importance sampling, MCMC and SMC) and fast approximations (Laplace and EP) are covered. Extensive numerical results are provided, some of which might go against conventional wisdom regarding the effectiveness of certain algorithms. Implications for other problems (variable selection) and other models are also discussed.

1. Introduction

The field of Bayesian computation seems hard to track these days, as it is blossoming in many directions. MCMC (Markov chain Monte Carlo) remains the main approach, but it is no longer restricted to Gibbs sampling and Hastings-Metropolis, as it includes more advanced, Physics-inspired methods, such as HMC (Hybrid Monte Carlo, Neal, 2010) and its variants (Girolami and Calderhead, 2011; Shahbaba et al., 2011; Hoffman and Gelman, 2013). On the other hand, there is also a growing interest for alternatives to MCMC, such as SMC (Sequential Monte Carlo, e.g. Del Moral et al., 2006), nested sampling (Skilling, 2006), or the fast approximations that originated from machine learning, such as Variational Bayes (e.g. Bishop, 2006, Chap. 10), and EP (Expectation Propagation, Minka, 2001). Even Laplace approximation has resurfaced in particular thanks to the INLA methodology (Rue et al., 2009).

One thing however that all these approaches have in common is they are almost always illustrated by a binary regression example; see e.g. the aforementioned papers. In other words, binary regressions models, such as probit or logit, are a de facto benchmark for Bayesian computation.

This remark leads to several questions. Are binary regression models a reasonable benchmark for Bayesian computation? Should they be used then to develop a 'benchmark culture' in Bayesian computation, like in e.g. optimisation? And practically, which of these methods actually 'works best' for approximating the posterior distribution of a binary regression model?

The objective of this paper is to answer these questions. As the ironic title suggests, our findings shall lead to us be critical of certain current practices. Specifically, most papers seem content with comparing some new algorithm with Gibbs sampling, on a few small datasets, such as the well-known Pima Indians diabetes dataset (8 covariates). But we shall see that, for such datasets, approaches that are even more basic than Gibbs sampling are actually hard to beat. In other words, datasets considered in the literature may be too toy-like to be used as a relevant benchmark. On the other hand, if ones considers larger datasets (with say 100 covariates), then not so many approaches seem to remain competitive.

We would also like to discuss *how* Bayesian computation algorithms should be compared. One obvious criterion is the error versus CPU time trade-off; this implies discussing which posterior quantities one may be need to approximate. A related point is whether the considered method comes with a simple way to evaluate the numerical error. Other criteria of interest are: (a) how easy to implement is the considered method? (b) how generic is it? (does changing the prior or the link function require a complete rewrite of the source code?) (c) to which extent does it require manual tuning to obtain good performances? (d) is it amenable to parallelisation? Points (a) and (b) are rarely discussed in Statistics, but relate to the important fact that, the simpler the program, the easier it is to maintain, and to make it bug-free. Regarding point (c), we warn beforehand that, as a matter of principle, we shall refuse to manually tune an algorithm on a per dataset basis. Rather, we will discuss, for each approach, some (hopefully reasonable) general recipe for how to choose the tuning parameters. This has two motivations. First, human time is far more valuable that computer time: Cook (2014) mentions that one hour of CPU time is today three orders of magnitude less expensive than one hour of pay for a programmer (or similarly a scientist). Second, any method requiring too much manual tuning through trial and error may be practically of no use beyond a small number of experts.

Finally, we also hope this paper may serve as an up to date review of the state of Bayesian computation. We believe this review to be timely for a number of reasons. First, as already mentioned, because Bayesian computation seems to develop currently in several different directions. Second, and this relates to criterion (d), the current interest in parallel computation (Lee et al., 2010; Suchard et al., 2010) may require a re-assessment of Bayesian computational methods: method A may perform better than method B on a single core architecture, while performing much worse on a parallel architecture. Finally, although the phrase 'big data' seems to be a tired trope already, it is certainly true that datasets are getting bigger and bigger, which in return means that statistical methods needs to be evaluated on bigger and bigger datasets. To be fair, we will not really consider in this work the kind of huge datasets that pertain to 'big data', but we will at least strive to move away from the kind of 'ridiculously small' data encountered too often in Bayesian computation papers.

The paper is structured as follows. Section 2 covers certain useful preliminaries on binary regression models. Section 3 discusses fast approximations, that is, deterministic algorithms that offer an approximation of the posterior, at a lower cost than sampling-based methods. Section 4 discusses 'exact', sampling-based methods. Section 5 is the most important part of the paper, as it contains an extensive numerical comparison of all these methods. Section 6 discusses variable selection. Section 7 discusses our findings, and their implications for both end users and Bayesian computation experts.

2. Preliminaries: binary regression models

2.1. **Likelihood, prior.** The likelihood of a binary regression model have the generic expression

$$p(\mathcal{D}|\boldsymbol{\beta}) = \prod_{i=1}^{n_{\mathcal{D}}} F(y_i \boldsymbol{\beta}^T \boldsymbol{x}_i) \tag{2.1}$$

where the data $\mathcal{D}$ consist of n responses $y_i \in \{-1, 1\}$ and n vectors $\boldsymbol{x}_i$ of p covariates, and F is some CDF (cumulative distribution function) that transforms the linear form $y_i \boldsymbol{\beta}^T \boldsymbol{x}_i$ into a probability. Taking $F = \Phi$, the standard normal CDF, gives the probit model, while taking $F = L$, the logistic CDF, $L(x) = 1/\left(1 + e^{-x}\right)$, leads

to the logistic model. Other choices could be considered, such as e.g. the CDF of a Student distribution (robit model) to better accommodate outliers.

We follow Gelman et al. (2008)'s recommendation to standardise the predictors in a preliminary step: non-binary predictors have mean 0 and standard deviation 0.5, binary predictors have mean 0 and range 1, and the intercept (if present) is set to 1. This standardisation facilitates prior specification: one then may set up a "weakly informative" prior for $\boldsymbol{\beta}$, that is a proper prior that assigns a low probability that the marginal effect of one predictor is outside a reasonable range. Specifically, we shall consider two priors $p(\boldsymbol{\beta})$ in this work: (a) the default prior recommended by Gelman et al. (2008), a product of independent Cauchys with centre 0 and scale 10 for the constant predictor, 2.5 for all the other predictors (henceforth, the Cauchy prior); and (b) a product of independent Gaussians with mean 0 and standard deviation equal to twice the scale of the Cauchy prior (henceforth the Gaussian prior).

Of course, other priors could be considered, such as e.g. Jeffreys' prior (Firth, 1993), or a Laplace prior (Kabán, 2007). Our main point in considering the two priors above is to determine to which extent certain Bayesian computation methods may be prior-dependent, either in their implementation (e.g. Gibbs sampling) or in their performance, or both. In particular, one may expect the Cauchy prior to be more difficult to deal with, given its heavy tails.

2.2. **Posterior maximisation (Gaussian prior).** We explain in this section how to quickly compute the mode, and the Hessian at the mode, of the posterior:

$$p(\boldsymbol{\beta}|\mathcal{D}) = \frac{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})}{p(\mathcal{D})}, \quad p(\mathcal{D}) = \int_{\mathbb{R}^d} p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})\, d\boldsymbol{\beta},$$

where $p(\boldsymbol{\beta})$ is one of the two priors presented in the previous section, and $Z(\mathcal{D})$ is the marginal likelihood of the data (also known as the evidence). These quantities will prove useful later, in particular to tune certain of the considered methods.

The two first derivatives of the log-posterior density may be computed as:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\beta}} \log p(\boldsymbol{\beta}|\mathcal{D}) &= \frac{\partial}{\partial \boldsymbol{\beta}} \log p(\boldsymbol{\beta}) + \frac{\partial}{\partial \boldsymbol{\beta}} \log p(\mathcal{D}|\boldsymbol{\beta}), \\
\frac{\partial^2}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^T} \log p(\boldsymbol{\beta}|\mathcal{D}) &= \frac{\partial^2}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^T} \log p(\boldsymbol{\beta}) + \frac{\partial^2}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^T} \log p(\mathcal{D}|\boldsymbol{\beta})
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\beta}} \log p(\mathcal{D}|\boldsymbol{\beta}) &= \sum_{i=1}^{n_\mathcal{D}} (\log F)' (y_i \boldsymbol{\beta}^T \boldsymbol{x}_i) y_i \boldsymbol{x}_i \\
\frac{\partial^2}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}^T} \log p(\mathcal{D}|\boldsymbol{\beta}) &= \sum_{i=1}^{n_\mathcal{D}} (\log F)'' (y_i \boldsymbol{\beta}^T \boldsymbol{x}_i) \boldsymbol{x}_i \boldsymbol{x}_i^T
\end{aligned}$$

and $(\log F)'$ and $(\log F)''$ are the two first derivatives of $\log F$. Provided that $\log F$ is concave, which is the case for probit and logit regressions, the Hessian of the log-likelihood is clearly a negative definite matrix. Moreover, if we consider the Gaussian prior, then the Hessian is of the log-posterior is also negative (as the sum of two negative matrices, as Gaussian densities are log-concave). We stick to the Gaussian prior for now.

This suggests the following standard approach to compute the MAP (maximum a posterior) estimator, that is the point $\boldsymbol{\beta}_{\text{MAP}}$ that maximises the posterior density $p(\boldsymbol{\beta}|\mathcal{D})$: to use Newton-Raphson, that is, to iterate

$$\boldsymbol{\beta}_{(\text{new})} = \boldsymbol{\beta}_{(\text{old})} - \boldsymbol{H}^{-1} \left\{ \frac{\partial}{\partial \boldsymbol{\beta}} \log p(\boldsymbol{\beta}_{(\text{old})}|\mathcal{D}) \right\} \tag{2.2}$$

until convergence is reached; here $\boldsymbol{H}$ is Hessian of the log posterior at $\boldsymbol{\beta} = \boldsymbol{\beta}_{(\text{old})}$, as computed above. The iteration above corresponds to finding the zero of a local, quadratic approximation of the log-posterior. Newton-Raphson typically works very well (converges in a small number of iterations) when the function to maximise is concave. A variant of this approach is

We note two points in passing. First, one may obtain the MLE (maximum likelihood estimator) by simply taking $p(\boldsymbol{\beta}) = 1$ above (i.e. a Gaussian with infinite variance). But the MLE is not properly defined when complete separation occurs, that is, there exists a hyperplane that separates perfectly the two outcomes: $y_i \boldsymbol{\beta}_{\text{CS}}^T \boldsymbol{x}_i \geq 0$ for some $\boldsymbol{\beta}_{\text{CS}}$ and all $i \in 1:N$. This remark gives an extra incentive for performing Bayesian inference, or at least MAP estimation, in cases where complete separation may occur, in particular when the number of covariates is large (Firth, 1993; Gelman et al., 2008).

Variants of Newton-Raphson may be obtained by adapting automatically the step size (e.g. update is $\boldsymbol{\beta}_{(\text{new})} = \boldsymbol{\beta}_{(\text{old})} - \lambda \boldsymbol{H}^{-1} \left\{ \frac{\partial}{\partial \boldsymbol{\beta}} \log p(\boldsymbol{\beta}_{(\text{old})}|\mathcal{D}) \right\}$, and step size λ is determined by line search) or replacing the Hessian $\boldsymbol{H}$ by some approximation. Some of these algorithms such as IRLS (iterated reweighted least squares) have a nice statistical interpretation. For our purposes however, these variants seem to show roughly similar performance, so we will stick to the standard version of Newton-Raphson.

2.3. **Posterior maximisation (Cauchy prior).** The log-density of the Cauchy prior is not concave:

$$\log p(\boldsymbol{\beta}) = -\sum_{j=1}^{p} \log(\pi \sigma_j) - \sum_{j=1}^{p} \log(1 + \beta_j^2/\sigma_j^2)$$

for scales σ_j chosen as explained in Section 2.1. Hence, the corresponding log-posterior is no longer guaranteed to be concave, which in turn means that Newton-Raphson might fail to converge.

However, we shall observe that, for most of the datasets considered in this paper, Newton-Raphson does converge quickly even for our Cauchy prior. In each case, we used as starting point for the Newton-Raphson iterations the OLS (ordinary least square) estimate. We suspect what happens is that, for most standard datasets, the posterior derived from a Cauchy prior remains log-concave, at least in a region that encloses the MAP estimator and our starting point.

3. Fast approximation methods

This section discusses fast approximation methods, that is methods that are deterministic, fast (compared to sampling-based methods), but which comes with an approximation error which is difficult to assess. These methods include the Laplace approximation, which was popular in Statistics before the advent of MCMC methods, but also recent Machine Learning methods, such as EP (Expectation Propagation, Minka, 2001), and VB (Variational Bayes, e.g. Bishop, 2006, Chap. 10). We will focus on Laplace and EP; for VB, see Consonni and Marin (2007) for a discussion of why VB (or at least a certain standard version of VB, known as mean field VB) may not work so well for probit models.

Concretely, we will focus on the approximation of the following posterior quantities: the marginal likelihood $p(\mathcal{D})$, as this may be used in model choice; and the marginal distributions $p(\beta_i|\mathcal{D})$ for each component β_i of $\boldsymbol{\beta}$. Clearly these are the

most commonly used summaries of the posterior distribution, and other quantities, such as the posterior expectation of $\boldsymbol{\beta}$, may be directly deduced from them.

Finally, one should bear in mind that such fast approximations may be used as a preliminary step to calibrate an exact, more expensive method, such as those described in Section 4.

3.1. **Laplace approximation.** The Laplace approximation is based on a Taylor expansion of the posterior log-density around the mode $\boldsymbol{\beta}_{\text{MAP}}$:

$$\log p(\boldsymbol{\beta}|\mathcal{D}) \approx \log p(\boldsymbol{\beta}_{\text{MAP}}|\mathcal{D}) - \frac{1}{2}\left(\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{MAP}}\right)^T \boldsymbol{Q}\left(\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{MAP}}\right),$$

where $\boldsymbol{Q} = -\boldsymbol{H}$, i.e. minus the Hessian of $\log p(\boldsymbol{\beta}|\mathcal{D})$ at $\boldsymbol{\beta} = \boldsymbol{\beta}_{\text{MAP}}$; recall that we explained how to compute these quantities in Section 2.2. One may deduce a Gaussian approximation of the posterior by simply exponentiating the equation above, and normalising:

$$\begin{aligned} q_L(\boldsymbol{\beta}) &= N_p\left(\boldsymbol{\beta}; \boldsymbol{\beta}_{\text{MAP}}, \boldsymbol{Q}^{-1}\right) \\ &:= (2\pi)^{-p/2}\left|\boldsymbol{Q}\right|^{1/2} \exp\left\{-\frac{1}{2}\left(\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{MAP}}\right)^T \boldsymbol{Q}\left(\boldsymbol{\beta}-\boldsymbol{\beta}_{\text{MAP}}\right)\right\}. \end{aligned} \tag{3.1}$$

In addition, since for any $\boldsymbol{\beta}$,

$$p(\mathcal{D}) = \frac{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})}{p(\boldsymbol{\beta}|\mathcal{D})}$$

one obtains an approximation to the marginal likelihood $p(\mathcal{D})$ as follows:

$$p(\mathcal{D}) \approx Z_L(\mathcal{D}) := \frac{p(\boldsymbol{\beta}_{\text{MAP}})p(\mathcal{D}|\boldsymbol{\beta}_{\text{MAP}})}{(2\pi)^{-p/2}\left|\boldsymbol{Q}\right|^{1/2}}.$$

From now on, we will refer to this particular Gaussian approximation q_L as the Laplace approximation, even if this phrase is sometimes used in Statistics for higher-order approximations, as discussed in the next Section. We defer to Section 3.5 the discussion of the advantages and drawbacks of this approximation scheme.

3.2. **Improved Laplace, connection with INLA.** Consider the marginal distributions $p(\beta_j|\mathcal{D}) = \int p(\boldsymbol{\beta}|\mathcal{D})\mathrm{d}\boldsymbol{\beta}_{-j}$ for each component β_j of $\boldsymbol{\beta}$, where $\boldsymbol{\beta}_{-j}$ is $\boldsymbol{\beta}$ minus β_j. A first approximation may be obtained by simply computing the marginals of the Laplace approximation q_L. An improved (but more expensive) approximation may be obtained from:

$$p(\beta_j|\mathcal{D}) \propto \frac{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})}{p(\boldsymbol{\beta}_{-j}|\beta_j,\mathcal{D})}$$

which suggests to choose a fine grid of β_j values (deduced for instance from $q_L(\boldsymbol{\beta})$), and for each β_j value, compute a Laplace approximation of $p(\boldsymbol{\beta}_{-j}|\beta_j,\mathcal{D})$, by computing the mode $\hat{\boldsymbol{\beta}}_{-j}(\beta_j)$ and the Hessian $\hat{\boldsymbol{H}}(\beta_j)$ of $\log p(\boldsymbol{\beta}_{-j}|\beta_j,\mathcal{D})$, and then approximate (up to a constant)

$$p(\beta_j|\mathcal{D}) \approx q_{IL}(\beta_j) \propto \frac{p\left(\hat{\boldsymbol{\beta}}(\beta_j)\right)p(\mathcal{D}|\hat{\boldsymbol{\beta}}(\beta_j))}{\left|\hat{H}(\beta_j)\right|^{1/2}}$$

where $\hat{\boldsymbol{\beta}}(\beta_j)$ is the vector obtained by inserting β_i at position i in $\hat{\boldsymbol{\beta}}_{-j}(\beta_j)$, and IL stands for "Improved Laplace". One may also deduce posterior expectations of functions of β_j in this way. See also Tierney and Kadane (1986), Tierney et al. (1989) for higher order approximations for posterior expectations.

We note in passing the connection to the INLA scheme of Rue et al. (2009). INLA applies to posteriors $p(\boldsymbol{\theta},\boldsymbol{x}|\mathcal{D})$ where $\boldsymbol{x}$ is a latent variable such that $p(\boldsymbol{x}|\boldsymbol{\theta},\mathcal{D})$ is

close to a Gaussian, and $\boldsymbol{\theta}$ is a low-dimensional hyper-parameter. It constructs a grid of $\boldsymbol{\theta}-$values, and for each grid point $\boldsymbol{\theta}_j$, it computes an improve Laplace approximation of the marginals of $p(\boldsymbol{x}|\boldsymbol{\theta}_j, \mathcal{D})$. In our context, $\boldsymbol{\beta}$ may be identified to $\boldsymbol{x}$, $\boldsymbol{\theta}$ to an empty set, and INLA reduces to the improved Laplace approximation described above.

3.3. **The EM algorithm of Gelman et al. (2008) (Cauchy prior).** Gelman et al. (2008) recommend against the Laplace approximation for a Student prior (of which our Cauchy prior is a special case), because, as explained in Section 2.3, the corresponding log-posterior is not guaranteed to be concave, and this might prevent Newton-Raphson to converge. In our simulations however, we found the Laplace approximation to work reasonably well for a Cauchy prior. We now briefly describe the alternative approximation scheme proposed by Gelman et al. (2008) for Student priors, which we call for convenience Laplace-EM.

Laplace-EM is based on the well-known representation of a Student distribution, $\beta_j|\sigma_j^2 \sim \mathrm{N}_1(0, \sigma_j^2)$, $\sigma_j^2 \sim \mathrm{Inv}-\mathrm{Gamma}(\nu/2, s_j\nu/2)$; take $\nu = 1$ to recover our Cauchy prior. Conditional on $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_p^2)$, the prior on $\boldsymbol{\beta}$ is Gaussian, hence, for a fixed $\boldsymbol{\sigma}^2$ one may implement Newton-Raphson to maximise the log-density of $p(\boldsymbol{\beta}|\boldsymbol{\sigma}^2, \mathcal{D})$, and deduce a Laplace (Gaussian) approximation of the same distribution.

Laplace-EM is an approximate EM (Expectation Maximisation, Dempster et al., 1977) algorithm, which aims at maximising in $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_p^2)$ the marginal posterior distribution $p(\boldsymbol{\sigma}^2|\mathcal{D}) = \int p(\boldsymbol{\sigma}^2, \boldsymbol{\beta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\beta}$. Each iteration involves an expectation with respect to the intractable conditional distribution $p(\boldsymbol{\beta}|\boldsymbol{\sigma}^2, \mathcal{D})$, which is Laplace approximated, using a single Newton-Raphson iteration. When this approximate EM algorithm has converged to some value $\boldsymbol{\sigma}_\star^2$, one more Newton-Raphson iteration is performed to compute a final Laplace approximation of $p(\boldsymbol{\beta}|\boldsymbol{\sigma}_\star^2, \mathcal{D})$, which is then reported as a Gaussian approximation to the posterior. We refer the readers to Gelman et al. (2008) for more details on Laplace-EM.

3.4. **Expectation-Propagation.** Like Laplace, Expectation Propagation (EP, Minka, 2001) generates a Gaussian approximation of the posterior, but it is based on different ideas. The consensus in machine learning seems to be that EP provides a better approximation than Laplace (e.g. Nickisch and Rasmussen, 2008); the intuition being that Laplace is 'too local' (i.e. it fitted so at to match closely the posterior around the mode), while EP is able to provide a global approximation to the posterior.

Starting from the decomposition of the posterior as product of $(n_{\mathcal{D}} + 1)$ factors:

$$p(\boldsymbol{\beta}|\mathcal{D}) = \frac{1}{p(\mathcal{D})} \prod_{i=0}^{n_{\mathcal{D}}} l_i(\boldsymbol{\beta}), \quad l_i(\boldsymbol{\beta}) = F(y_i \boldsymbol{\beta}^T \boldsymbol{x}_i) \text{ for } i \geq 1,$$

and l_0 is the prior, $l_0(\boldsymbol{\beta}) = p(\boldsymbol{\beta})$, EP computes iteratively a parametric approximation of the posterior with the same structure

$$q_{\mathrm{EP}}(\boldsymbol{\beta}) = \prod_{i=0}^{n_{\mathcal{D}}} \frac{1}{Z_i} q_i(\boldsymbol{\beta}). \tag{3.2}$$

Taking q_i to be an unnormalised Gaussian densities written in natural exponential form

$$q_i(\boldsymbol{\beta}) = \exp\left\{ -\frac{1}{2} \boldsymbol{\beta}^T \boldsymbol{Q}_i \boldsymbol{\beta} + \boldsymbol{\beta}^T \boldsymbol{r}_i \right\},$$

one obtains for q_{EP} a Gaussian with natural parameters $\boldsymbol{Q} = \sum_{i=0}^{n} \boldsymbol{Q}_i$ and $\boldsymbol{r}_i = \sum_{i=0}^{n} \boldsymbol{r}_i$; note that the more standard parametrisation of Gaussians may be recovered by taking

$$\boldsymbol{\Sigma} = \boldsymbol{Q}^{-1}, \quad \boldsymbol{\mu} = \boldsymbol{Q}^{-1}\boldsymbol{r}.$$

Other exponential families could be considered for q and the q_i's, see e.g. Seeger (2005), but Gaussian approximations seems the most natural choice here.

An EP iteration consists in updating one factor q_i, or equivalently $(Z_i, \boldsymbol{Q}_i, \boldsymbol{r}_i)$, while keeping the other factors as fixed, by moment matching between the hybrid distribution

$$h(\boldsymbol{\beta}) \propto l_i(\boldsymbol{\beta}) \prod_{j \neq i} q_j(\boldsymbol{\beta})$$

and the global approximation q defined in (3.2): compute

$$\begin{aligned} Z_h &= \int l_i(\boldsymbol{\beta}) \prod_{j \neq i} q_j(\boldsymbol{\beta}) \, \mathrm{d}\boldsymbol{\beta} \\ \boldsymbol{\mu}_h &= \frac{1}{Z_h} \int \boldsymbol{\beta} l_i(\boldsymbol{\beta}) \prod_{j \neq i} q_j(\boldsymbol{\beta}) \, \mathrm{d}\boldsymbol{\beta} \\ \boldsymbol{\Sigma}_h &= \frac{1}{Z_h} \int \boldsymbol{\beta}\boldsymbol{\beta}^T l_i(\boldsymbol{\beta}) \prod_{j \neq i} q_j(\boldsymbol{\beta}) \, \mathrm{d}\boldsymbol{\beta} \end{aligned}$$

and set

$$\boldsymbol{Q}_i = \boldsymbol{\Sigma}_h^{-1} - \boldsymbol{Q}_{-i}, \quad \boldsymbol{r}_i = \boldsymbol{\Sigma}_h^{-1}\boldsymbol{\mu}_h - \boldsymbol{r}_{-i}, \quad \log Z_i = \log Z_h - \Psi(\boldsymbol{r}, \boldsymbol{Q}) + \Psi(\boldsymbol{r}_{-i}, \boldsymbol{Q}_{-i})$$

where $\boldsymbol{r}_{-i} = \sum_{j \neq i} \boldsymbol{r}_j$, $\boldsymbol{Q}_{-i} = \sum_{j \neq i} \boldsymbol{Q}_j$, and $\psi(\boldsymbol{r}, \boldsymbol{Q})$ is the normalising constant of a Gaussian distribution with natural parameters $(\boldsymbol{r}, \boldsymbol{Q})$,

$$\psi(\boldsymbol{r}, \boldsymbol{Q}) = \int_{\mathbb{R}^p} \exp\left\{ -\frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{Q}\boldsymbol{\beta} + \boldsymbol{\beta}^T \boldsymbol{r} \right\} \mathrm{d}\boldsymbol{\beta} = -\frac{1}{2} \log |\boldsymbol{Q}/2\pi| + \frac{1}{2}\boldsymbol{r}^T \boldsymbol{Q}\boldsymbol{r}.$$

In practice, EP proceeds by looping over sites, updating each one in turn until convergence is achieved.

To implement EP for binary regression models, two points must be addressed. First, how to compute the hybrid moments? For the probit model, these moments may be computed exactly, see the supplement, while for the other links function (such as logistic), numerical (one-dimensional) quadrature may be used. Second, how to deal with the prior? If the prior is Gaussian, one may simply set q_0 to the prior, and never update q_0 in the course of the algorithm. For a Cauchy prior, q_0 is simply treated as an extra site.

EP being a fairly recent method, it is currently lacking in terms of supporting theory, both in terms of algorithmic convergence (does it converge in a finite number of iterations?), and statistical convergence (does the resulting approximation converges in some sense to the true posterior distribution as $n_{\mathcal{D}} \to +\infty$?). On the other hand, there is mounting evidence that EP works very well in many problems; again see e.g. Nickisch and Rasmussen (e.g. 2008).

3.5. **Discussion of the different approximation schemes.** Laplace and its variants have complexity $\mathcal{O}(n_{\mathcal{D}} + p^3)$, while EP has complexity $\mathcal{O}(n_{\mathcal{D}} p^3)$. Incidentally, one sees that the number of covariates p is more critical than the number of instances $n_{\mathcal{D}}$ in determining how 'big' (how time-intensive to process) is a given dataset. This will be a recurring point in this paper.

The p^3 term in both complexities is due to the $p \times p$ matrix operations performed by both algorithms; e.g. the Newton-Raphson update (2.2) requires solving a linear system of order p. EP requires to perform such p^3 operations at each site (i.e. for

each single observation), hence the $\mathcal{O}(n_{\mathcal{D}}p^3)$ complexity, while Laplace perform such operations only once per iteration. EP is therefore expected to be more expensive than Laplace.

This remark may be mitigated as follows. First, one may modify EP so as to update the global approximation only at the end of each iteration (complete pass over the data). The resulting algorithm (van Gerven et al., 2010) may be easily implemented on parallel hardware: simply distribute the $n_{\mathcal{D}}$ factors over the processors. Even without parallelisation, parallel EP requires only one single matrix inversion per iteration.

Second, the 'improved Laplace' approximation for the marginals described in Section 3.1 requires to perform quite a few basic Laplace approximations, so its speed advantage compared to standard EP essentially vanishes.

Points that remain in favour of Laplace is that it is simpler to implement than EP, and the resulting code is very generic: adapting to either a different prior, or a different link function (choice of F in 2.1), is simply a matter of writing a function that evaluates the corresponding function. We have seen that such an adaptation requires more work in EP, although to be fair the general structure of the algorithm is not model-dependent. On the other hand, we shall see that EP is often more accurate, and works in more examples, than Laplace; this is especially the case for the Cauchy prior.

4. Exact methods

We now turn to sampling-based methods, which are 'exact', at least in the limit: one may make the approximation error as small as desired, by running the corresponding algorithm for long enough. We will see that all of these algorithms requires some form of calibration that requires prior knowledge on the shape of the posterior distribution. Since the approximation methods covered in the previous section are faster by orders of magnitude than sampling-based methods, we will assume that a Gaussian approximation $q(\boldsymbol{\beta})$ (say, obtained by Laplace or EP) has been computed in a preliminary step.

4.1. Our gold standard: Importance sampling.

Let $q(\boldsymbol{\beta})$ denote a generic approximation of the posterior $p(\boldsymbol{\beta}|\mathcal{D})$. Importance sampling (IS) is based on the trivial identity

$$p(\mathcal{D}) = \int p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})\,\mathrm{d}\boldsymbol{\beta} = \int q(\boldsymbol{\beta})\frac{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})}{q(\boldsymbol{\beta})}\,\mathrm{d}\boldsymbol{\beta}$$

which leads to the following recipe: sample $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N \sim q$, then compute as an estimator of $p(\mathcal{D})$

$$Z_N = \frac{1}{N}\sum_{n=1}^{N} w(\boldsymbol{\beta}_n), \quad w(\boldsymbol{\beta}) := \frac{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})}{q(\boldsymbol{\beta})}. \tag{4.1}$$

In addition, since

$$\int \varphi(\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\beta} = \frac{\int \varphi(\boldsymbol{\beta})q(\boldsymbol{\beta})w(\boldsymbol{\beta})\,\mathrm{d}\boldsymbol{\beta}}{\int q(\boldsymbol{\beta})w(\boldsymbol{\beta})\,\mathrm{d}\boldsymbol{\beta}}$$

one may approximate any posterior moment as

$$\varphi_N = \frac{\sum_{n=1}^{N} w(\boldsymbol{\beta}_n)\varphi(\boldsymbol{\beta}_n)}{\sum_{n=1}^{N} w(\boldsymbol{\beta}_n)}. \tag{4.2}$$

Approximating posterior marginals is also straightforward; one may for instance use kernel density estimation on the weighted sample $(\boldsymbol{\beta}_n, w(\boldsymbol{\beta}_n))_{n=1}^{N}$.

Concerning the choice of q, we will restrict ourselves to the Gaussian approximations generated either from Laplace or EP algorithm. It is sometimes recommended to use a Student distribution instead, as a way to ensure that the variance of the above estimators is finite, but we did not observe any benefit for doing so in our simulations.

It is of course a bit provocative to call IS our gold standard, as it is sometimes perceived as an obsolete method. We would like to stress out however that IS is hard to beat relative to most of the criteria laid out in the introduction:

- because it is based on IID sampling, assessing the Monte Carlo error of the above estimators is trivial: e.g. the variance of Z_N may be estimated as N^{-1} times the empirical variance of the weights $w(\boldsymbol{\beta}_n)$. The auto-normalised estimator 4.2 has asymptotic variance
$$\mathbb{E}_q\left[w(\boldsymbol{\beta})^2\left\{\varphi(\boldsymbol{\beta})-\mu(\varphi)\right\}^2\right], \quad \mu(\varphi)=\int \varphi(\boldsymbol{\beta})p(\boldsymbol{\beta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\beta}$$
which is also trivial to approximate from the simulated $\boldsymbol{\beta}_n$'s.
- Other advantages brought by IID sampling are: (a) importance sampling is easy to parallelize; and (b) importance sampling is amenable to QMC (Quasi-Monte Carlo) integration, as explained in the following section.
- Importance sampling offers an approximation of the marginal likelihood $p(\mathcal{D})$ at no extra cost.
- Code is simple and generic.

Of course, what remains to determine is whether importance sampling does well relative to our main criterion, i.e. error versus CPU trade-off. We do know that IS suffers from a curse of dimensionality: take both q and and the target density π to be the density of IID distributions: $q(\boldsymbol{\beta})=\prod_{j=1}^p q_1(\beta_j)$, $\pi(\boldsymbol{\beta})=\prod_{j=1}^p \pi_1(\beta_j)$; then it is easy to see that the variance of the weights grows exponentially with p. Thus we expect IS to collapse when p is too large; meaning that a large proportion of the $\boldsymbol{\beta}_n$ gets a negligible weight. On the other hand, for small to moderate dimensions, we will observe surprising good results; see Section 5. We will also present below a SMC algorithm that automatically reduces to IS when IS performs well, while doing something more elaborate in more difficult scenarios.

The standard way to assess the weight degeneracy is to compute the effective sample size (Kong et al., 1994),
$$\mathrm{ESS}=\frac{\left\{\sum_{n=1}^N w(\boldsymbol{\beta}_n)\right\}^2}{\sum_{n=1}^N w(\boldsymbol{\beta}_n)^2}\in[1,N],$$
which roughly approximates how many simulations from the target distribution would be required to produce the same level of error. In our simulations, we will compute instead the efficiency factor EF, which is simply the ratio $\mathrm{EF}=\mathrm{ESS}/N$.

4.2. **Improving importance sampling by Quasi-Monte Carlo.** Quasi-Monte Carlo may be seen as an elaborate variance reduction technique: starting from the Monte Carlo estimators Z_N and φ_N, see (4.1) and (4.2), one may re-express the simulated vectors as functions of uniform variates $\boldsymbol{u}_n$ in $[0,1]^d$; for instance:
$$\boldsymbol{\beta}_n=\boldsymbol{\mu}+\boldsymbol{C}\boldsymbol{\zeta}_n, \quad \boldsymbol{\zeta}_n=\boldsymbol{\Phi^{-1}}(\boldsymbol{u}_n)$$
where $\boldsymbol{\Phi^{-1}}$ is Φ^{-1}, the $N(0,1)$ inverse CDF, applied component-wise. Then, one replaces the N vectors $\boldsymbol{u}_n$ by a low-discrepancy sequence; that is a sequence of N vectors that spread more evenly over $[0,1]^d$; e.g. a Halton or a Sobol' sequence. Under appropriate conditions, QMC error converges at rate $\mathcal{O}(N^{-1+\epsilon})$, for any $\epsilon>0$, to be compared with the standard Monte Carlo rate $\mathcal{O}_P(N^{-1/2})$. We refer

to Lemieux (2009) for more background on QMC, as well as how to construct QMC sequences.

Oddly enough, the possibility to use QMC in conjunction with importance sampling is very rarely mentioned in the literature; see however Hörmann and Leydold (2005). More generally, QMC seems often overlooked in Statistics. We shall see however that this simple IS-QMC strategy often performs very well.

One drawback of IS-QMC is that we lose the ability to evaluate the approximation error in a simple manner. A partial remedy is to use randomised Quasi-Monte Carlo (RQMC), that is, the $\boldsymbol{u}_n$ are generated in such a way that (a) with probability one, $\boldsymbol{u}_{1:N}$ is a QMC point set; and (b) each vector $\boldsymbol{u}_n$ is marginally sampled from $[0,1]^d$. Then QMC estimators that are empirical averages, such as $Z_N = N^{-1}\sum_{n=1}^N w(\boldsymbol{\beta}_n)$ become unbiased estimators, and their error may be assessed through the empirical variance over repeated runs. Technically, estimators that are ratios of QMC averages, such as φ_N, are not unbiased, but for all practical purposes their bias is small enough that assessing error through empirical variances over repeated runs remains a reasonable approach.

4.3. **MCMC.** The general principle of MCMC (Markov chain Monte Carlo) is to simulate a Markov chain that leaves invariant the posterior distribution $p(\boldsymbol{\beta}|\mathcal{D})$; see Robert and Casella (2004) for a general overview. Often mentioned drawbacks of MCMC simulation are (a) the difficulty to parallelize such algorithms (although see e.g. Jacob et al., 2011 for an attempt at this problem); (b) the need to specify a good starting point for the chain (or alternatively to determine the burn-in period, that is, the length of the initial part of the chain that should be discarded) and (c) the difficulty to assess the convergence of the chain (that is, to determine if the distribution of $\boldsymbol{\beta}_t$ at iteration t is sufficiently close to the invariant distribution $p(\boldsymbol{\beta}|\mathcal{D})$).

To be fair, these problems are not so critical for binary regression models. Regarding (b), one may simply start the chain from the posterior mode, or from a draw of one of the Gaussian approximations covered in the previous section. Regarding (c) for most standard datasets, MCMC converges reasonably fast, and convergence is easy to assess visually. The main issue in practice is that MCMC generates correlated random variables, and these correlations inflate the Monte Carlo variance.

4.3.1. *Gibbs sampling.* Consider the following data-augmentation formulation of binary regression:

$$\begin{aligned} z_i &= \boldsymbol{\beta}^T \boldsymbol{x}_i + \epsilon_i \\ y_i &= \mathrm{sgn}(z_i) \end{aligned}$$

where $\boldsymbol{z} = (z_1, \ldots, z_{n_\mathcal{D}})^T$ is a vector of latent variables, and assume for a start that $\epsilon_i \sim N(0,1)$ (probit regression). One recognises $p(\boldsymbol{\beta}|\boldsymbol{z},\mathcal{D})$ as the posterior of a linear regression model, which is tractable (for an appropriate prior). This suggests to sample from $p(\boldsymbol{\beta},\boldsymbol{z}|\mathcal{D})$ using Gibbs sampling (Albert and Chib, 1993): i.e. iterate the two following steps: (a) sample from $\boldsymbol{z}|\boldsymbol{\beta},\mathcal{D}$; and (b) sample from $\boldsymbol{\beta}|\boldsymbol{z},\mathcal{D}$.

For (a), the z_i's are conditionally independent, and follows a truncated Gaussian distribution

$$p(z_i|\boldsymbol{\beta},\mathcal{D}) \propto \mathrm{N}_1\left(z_i; \boldsymbol{\beta}^T\boldsymbol{x}_i, 1\right) \mathbb{1}\left\{z_i y_i > 0\right\}$$

which is easy to sample from (Chopin, 2011). For Step (b) and a Gaussian prior $\mathrm{N}_p(\mathbf{0}, \boldsymbol{\Sigma}_{\mathrm{prior}})$, one has, thanks to standard conjugacy properties:

$$\boldsymbol{\beta}|\boldsymbol{z},\mathcal{D} \sim \mathrm{N}_p\left(\boldsymbol{\mu}_{\mathrm{post}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\mathrm{post}}\right), \quad \boldsymbol{\Sigma}_{\mathrm{post}}^{-1} = \boldsymbol{\Sigma}_{\mathrm{prior}}^{-1} + \boldsymbol{x}\boldsymbol{x}^T, \quad \boldsymbol{\mu}_{\mathrm{post}}(\boldsymbol{z}) = \boldsymbol{\Sigma}_{\mathrm{post}}^{-1}\boldsymbol{x}\boldsymbol{z}$$

Algorithm 1 Hastings-Metropolis iteration

Input: $\boldsymbol{\beta}$
Output: $\boldsymbol{\beta}'$
1: Sample $\boldsymbol{\beta}^\star \sim \kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta})$.
2: With probability $1 \wedge r$,

$$r = \frac{p(\boldsymbol{\beta}^\star)p(\mathcal{D}|\boldsymbol{\beta}^\star)\kappa(\boldsymbol{\beta}|\boldsymbol{\beta}^\star)}{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta})},$$

set $\boldsymbol{\beta}' = \boldsymbol{\beta}^\star$; otherwise set $\boldsymbol{\beta}' = \boldsymbol{\beta}$.

where $\boldsymbol{x}$ is the $n \times p$ matrix obtained by stacking the $\boldsymbol{x}_i^T$. Note that $\boldsymbol{\Sigma}_{\text{post}}$ and its inverse need to be computed only once, hence the complexity of a Gibbs iteration is $\mathcal{O}(p^2)$, not $\mathcal{O}(p^3)$.

The main drawback of Gibbs sampling is that it is particularly not generic: its implementation depends very strongly on the prior and the model. Sticking to the probit case, switching to another prior requires deriving a new way to update $\boldsymbol{\beta}|\boldsymbol{z}, \mathcal{D}$. For instance, for a prior which is a product of Students with scales σ_j (e.g. our Cauchy prior), one may add extra latent variables, by resorting to the well-known representation: $\beta_j|s_j \sim \mathrm{N}_1(0, \nu\sigma_j^2/s_j)$, $s_j \sim \mathrm{Chi}^2(\nu)$; with $\nu = 1$ for our Cauchy prior. Then the algorithm has three steps: (a) an update of the z_i's, exactly as above; (b) an update of $\boldsymbol{\beta}$, as above but with $\boldsymbol{\Sigma}_{\text{prior}}$ replaced by the diagonal matrix with elements $\nu\sigma_j^2/s_j$, $j = 1, \ldots, p$; and (c) an (independent) update of the p latent variables s_j, with $s_j|\boldsymbol{\beta}, \boldsymbol{z}, \mathcal{D} \sim \text{Gamma}\left((1+\nu)/2, \left(1 + \nu\beta_j^2/\sigma_j^2\right)/2\right)$. The complexity of Step (b) is now $\mathcal{O}(p^3)$, since $\boldsymbol{\Sigma}_{\text{prior}}$ and $\boldsymbol{\Sigma}_{\text{post}}$ must be recomputed at each iteration (although some speed-up may be obtained by using Sherman–Morrison formula).

Of course, considering yet another type of prior would require deriving another strategy for sampling $\boldsymbol{\beta}$. Then if one turns to logistic regression, things get rather complicated. In fact, deriving an efficient Gibbs sampler for logistic regression is a topic of current research; see Holmes and Held (2006); Frühwirth-Schnatter and Frühwirth (2009); Gramacy and Polson (2012); Polson et al. (2013). In a nutshell, the two first papers use the same data augmentation as above, but with $\epsilon_i \sim$ Logistic(1) written as a certain mixture of Gaussians (infinite for the first paper, finite but approximate for the second paper), while Polson et al. (2013) use instead a representation of a logistic likelihood as an infinite mixture of Gaussians, with a Polya-Gamma as the mixing distribution. Each representation leads to introducing extra latent variables, and discussing how to sample their conditional distributions.

Since their implementation is so model-dependent, the main justification for Gibbs samplers should be their greater performance relative to more generic algorithms. We will investigate if this is indeed the case in our numerical section.

4.3.2. *Hastings-Metropolis.* Hastings-Metropolis consists in iterating the step described as Algorithm 1. Much like importance sampling, Hastings-Metropolis is both simple and generic, that is, up to the choice of the proposal kernel $\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta})$ (the distribution of the proposed point $\boldsymbol{\beta}^\star$, given the current point $\boldsymbol{\beta}$). A naive approach is to take $\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta})$ independent of $\boldsymbol{\beta}$, $\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta}) = q(\boldsymbol{\beta}^\star)$, where q is some approximation of the posterior. In practice, this usually does not work better than importance sampling based on the same proposal, hence this strategy is hardly used.

Algorithm 2 Leap-frog step

Input: $(\boldsymbol{\beta}, \boldsymbol{\alpha})$
Output: $(\boldsymbol{\beta}_1, \boldsymbol{\alpha}_1)$
1: $\boldsymbol{\alpha}_{1/2} \leftarrow \boldsymbol{\alpha} - \frac{\epsilon}{2}\nabla_{\boldsymbol{\beta}} E(\boldsymbol{\beta})$
2: $\boldsymbol{\beta}_1 \leftarrow \boldsymbol{\beta} + \epsilon \boldsymbol{\alpha}_{1/2}$
3: $\boldsymbol{\alpha}_1 \leftarrow \boldsymbol{\alpha}_{1/2} - \frac{\epsilon}{2}\nabla_{\boldsymbol{\beta}} E(\boldsymbol{\beta}_1)$

A more usual strategy is to set the proposal kernel to a random walk: $\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta}) = \mathrm{N}_p(\boldsymbol{\beta}, \boldsymbol{\Sigma}_{\text{prop}})$. It is well known that the choice of $\boldsymbol{\Sigma}_{\text{prop}}$ is critical for good performance. For instance, in the univariate case, if $\boldsymbol{\Sigma}_{\text{prop}}$ is too small, the chain moves slowly, while if too large, proposed moves are rarely accepted.

A result from the optimal scaling literature (e.g. Roberts and Rosenthal, 2001) is that, for a $\mathrm{N}_p(\mathbf{0}, \boldsymbol{I}_p)$ target, $\boldsymbol{\Sigma}_{\text{prop}} = (\lambda^2/p)\boldsymbol{I}_p$ with $\lambda = 2.38$ is asymptotically optimal, in the sense that as $p \to \infty$, this choice leads to the fastest exploration. Since the posterior of a binary regression model is reasonably close to a Gaussian, we adapt this result by taking $\boldsymbol{\Sigma}_{\text{prop}} = (\lambda^2/p)\boldsymbol{\Sigma}_q$ in our simulations, where $\boldsymbol{\Sigma}_q$ is the covariance matrix of a (Laplace or EP) Gaussian approximation of the posterior. This strategy seems validated by the fact we obtain acceptance rates close to the optimal rate, as given by Roberts and Rosenthal (2001).

The bad news behind this optimality result is that the chain requires $\mathcal{O}(p)$ steps to move a $\mathcal{O}(1)$ distance. Thus random walk exploration tends to become slow for large p. This is usually cited as the main motivation to develop more elaborate MCMC strategies, such as HMC, which we cover in the following section.

4.3.3. *HMC.* Hamiltonian Monte Carlo (HMC, also known as Hybrid Monte Carlo, Duane et al., 1987) is a new type of MCMC algorithm, where one is able to perform several steps in the parameter space before determining if the new position is accepted or not. Consequently, HMC is able to make much bigger jumps in the parameter space than standard Metropolis algorithms. See Neal (2010) for an excellent introduction.

Consider the pair $(\boldsymbol{\beta}, \boldsymbol{\alpha})$, where $\boldsymbol{\beta} \sim p(\boldsymbol{\beta}|\mathcal{D})$, and $\boldsymbol{\alpha} \sim N_p(0, M^{-1})$, thus with joint un-normalised density $\exp\{-H(\boldsymbol{\beta}, \boldsymbol{\alpha})\}$, with

$$H(\boldsymbol{\beta}, \boldsymbol{\alpha}) = E(\boldsymbol{\beta}) + \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{M} \boldsymbol{\alpha}, \quad E(\boldsymbol{\beta}) = -\log\{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})\}.$$

The physical interpretation of HMC is that of a particle at position $\boldsymbol{\beta}$, with velocity $\boldsymbol{\alpha}$, potential energy $E(\boldsymbol{\beta})$, kinetic energy $\frac{1}{2}\boldsymbol{\alpha}^T M \boldsymbol{\alpha}$, for some mass matrix M, and therefore total energy given by $H(\boldsymbol{\beta}, \boldsymbol{\alpha})$. The particle is expected to follow a trajectory such that $H(\boldsymbol{\beta}, \boldsymbol{\alpha})$ remains constant over time.

In practice, HMC proceeds as follows: first, sample a new velocity vector, $\boldsymbol{\alpha} \sim N_p(0, M^{-1})$. Second, move the particle while keeping the Hamiltonian H constant; in practice, discretisation must be used, so L steps of step-size ϵ are performed through leap-frop steps; see Algorithm 2 which describes one such step. Third, the new position, obtained after L leap-frog steps is accepted or rejected according to probability $1 \wedge \exp\{H(\boldsymbol{\beta}, \boldsymbol{\alpha}) - H(\boldsymbol{\beta}^\star, \boldsymbol{\alpha}^\star)\}$; see Algorithm 3 for a summary. The validity of the algorithm relies on the fact that a leap-frog step is "volume preserving"; that is, the deterministic transformation $(\boldsymbol{\beta}, \boldsymbol{\alpha}) \to (\boldsymbol{\beta}_1, \boldsymbol{\alpha}_1)$ has Jacobian one. This is why the acceptance probability admits this simple expression.

The tuning parameters of HMC are $\boldsymbol{M}$ (the mass matrix), L (number of leap-frog steps), and ϵ (the stepsize). For $\boldsymbol{M}$, we follow Neal (2010)'s recommendation and take $\boldsymbol{M}^{-1} = \boldsymbol{\Sigma}_q$, an approximation of the posterior variance (again obtained from either Laplace or EP). This is equivalent to rescaling the posterior so as to

Algorithm 3 HMC iteration

Input: $\boldsymbol{\beta}$
Output: $\boldsymbol{\beta}'$
1: Sample momentum $\boldsymbol{\alpha} \sim \mathrm{N}_p(0, \boldsymbol{M})$.
2: Perform L leap-frog steps (see Algorithm 2), starting from $(\boldsymbol{\beta}, \boldsymbol{\alpha})$; call $(\boldsymbol{\beta}^\star, \boldsymbol{\alpha}^\star)$ the final position.
3: With probability $1 \wedge r$,

$$r = \exp\left\{H(\boldsymbol{\beta}, \boldsymbol{\alpha}) - H(\boldsymbol{\beta}^\star, \boldsymbol{\alpha}^\star)\right\}$$

set $\boldsymbol{\beta}^{'} = \boldsymbol{\beta}^\star$; otherwise set $\boldsymbol{\beta}^{'} = \boldsymbol{\beta}$.

have a covariance matrix close to identity. In this way, we avoid the bad mixing typically incurred by strong correlations between components.

The difficulty to choose L and ϵ seems to be the main drawback of HMC. The performance of HMC seems very sensitive to these tuning parameters, yet clear guidelines on how to choose them seem currently lacking. A popular approach is to fix $L\epsilon$ to some value, and to use vanishing adaptation (Andrieu and Thoms, 2008) to adapt ϵ so as to target acceptance rate of 0.65 (the optimal rate according to the formal study of HMC by Beskos et al., 2013): i.e. at iteration t, take $\epsilon = \epsilon_t$, with $\epsilon_t = \epsilon_{t-1} - \eta_t(R_t - 0.65)$, $\eta_t = t^{-\kappa}$, $\kappa \in (1/2, 1)$ and R_t the acceptance rate up to iteration t. The rationale for fixing $L\epsilon$ is that quantity may be interpreted as a 'simulation length', i.e. how much distance one moves at each step; if too small, the algorithm may exhibit random walk behaviour, while if too large, it may move a long distance before coming back close to its starting point. Since the spread of is already taken into account through $\boldsymbol{M}^{-1} = \boldsymbol{\Sigma}_q$, we took $\epsilon L = 1$ in our simulations.

4.3.4. *NUTS and other variants of HMC.* Girolami and Calderhead (2011) proposed an interesting variation of HMC, where the mass matrix $\boldsymbol{M}$ is allowed to depends on $\boldsymbol{\beta}$; e.g. $\boldsymbol{M}(\boldsymbol{\beta})$ is set to the Fisher information of the model. This allows the corresponding algorithm, called RHMC (Riemanian HMC), to adapt locally to the geometry of the target distribution. The main drawback of RHMC is that each iteration involves computing derivatives of $M(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$, which is very expensive, especially if p is large. For binary regression, we found RMHC to be too expensive relative to plain HMC, even when taking into account the better exploration brought by RHMC. This might be related to the fact that the posterior of a binary regression model is rather Gaussian-like and thus may not require such a local adaptation of the sampler.

We now focus on NUTS (No U-Turn sampler, Hoffman and Gelman, 2013), a variant of HMC which does not require to specify a priori L, the number of leap-frog steps. Instead, NUTS aims at keeping on doing such steps until the trajectory starts to loop back to its initial position. Of course, the difficulty in this exercise is to preserve the time reversibility of the simulated Markov chain. To that effect, NUTS constructs iteratively a binary tree whose leaves correspond to different velocity-position pairs $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ obtained after a certain number of leap-frog steps. The tree starts with two leaves, one at the current velocity-position pair, and another leaf that corresponds to one leap-frop step, either in the forward or backward direction (i.e. by reversing the sign of velocity); then it iteratively doubles the number of leaves, by taking twice more leap frog steps, again either in the forward or backward direction. The tree stops growing when at least one leaf corresponds to a "U-turn"; then NUTS chooses randomly one leaf, among those leaves that would have generated the current position with the same binary tree

mechanism; in this way reversibility is preserved. Finally NUTS moves the new position that corresponds to the chosen leaf.

We refer the readers to Hoffman and Gelman (2013) for a more precise description of NUTS. Given its complexity, implementing directly NUTS seems to require more efforts than the other algorithms covered in this paper. Fortunately, the STAN package (`http://mc-stan.org/`) provides a C++ implementation of NUTS which is both efficient and user-friendly: the only required input is a description of the model in a probabilistic programming language similar to BUGS. In particular, STAN is able to automatically derive the log-likelihood and its gradient, and no tuning of any sort is required from the user. Thus, we will use STAN to assess NUTS in our numerical comparisons.

4.4. **Sequential Monte Carlo.** Sequential Monte Carlo (SMC) is a class of algorithms for approximating iteratively a sequence of distributions π_t, $t = 0, \ldots, T$, using importance sampling, resampling, and MCMC steps. We focus here on the non-sequential use of SMC (Neal, 2001; Chopin, 2002; Del Moral et al., 2006), where one is only interested in approximating the final distribution π_T (in our case, set to the posterior $p(\boldsymbol{\beta}|\mathcal{D})$), and the previous π_t's are designed so as to allow for a smooth progression from some π_0, which is easy to sample from, to π_T.

At iteration t, SMC produces a set of weighted particles (simulations) $(\boldsymbol{\beta}_n, w_n)_{n=1}^N$ that approximates π_t, in the sense that

$$\frac{1}{\sum_{n=1}^N w_n} \sum_{n=1}^N w_n \varphi(\boldsymbol{\beta}_n) \to \mathbb{E}^{\pi_t}\left[\varphi(\boldsymbol{\beta})\right]$$

as $N \to +\infty$. At time 0, one samples $\boldsymbol{\beta}^n \sim \pi_0$, and set $w_n = 1$. To progress from π_{t-1} to π_t, one uses importance sampling: weights are multiplied by ratio $\pi_t(\boldsymbol{\beta}_n)/\pi_{t-1}(\boldsymbol{\beta}_n)$. When the variance of the weights gets too large (which indicates that too few particles contribute significantly to the current approximation), one resamples the particles: each particle gets reproduced O_n times, where $O_n \geq 0$ is random, and such that $\mathbb{E}(O_n) = N w_n / \sum_{m=1}^N w_m$, and $\sum_{n=1}^N O_n = N$ with probability one. In this way, particles with a low weights are likely to die, while particles with a large weight get reproduced many times. Finally, one may re-introduce diversity among the particles by applying one (or several) MCMC steps, using a MCMC kernel that leaves invariant the current distribution π_t.

We focus in this paper on tempering SMC, where the sequence

$$\pi_t(\boldsymbol{\beta}) \propto q(\boldsymbol{\beta})^{1-\delta_t} \left\{p(\boldsymbol{\beta}) p(\mathcal{D}|\boldsymbol{\beta})\right\}^{\delta_t}$$

corresponds to a linear interpolation (on the log-scale) between some distribution $\pi_0 = q$, and $\pi_T(\boldsymbol{\beta}) = p(\boldsymbol{\beta}|\mathcal{D})$, our posterior. This is a convenient choice in our case, as we have at our disposal some good approximation q (either from Laplace or EP) of our posterior. A second advantage of tempering SMC is that one can automatically adapt the "temperature ladder" δ_t (Jasra et al., 2011). Algorithm 4 describes a tempering SMC algorithm based on such an adaptation scheme: at each iteration, the next distribution π_t is chosen so that the efficiency factor (defined in Section 4.1) of the importance sampling step from π_{t-1} to π_t equals a pre-defined level $\tau \in (0, 1)$; a default value is $\tau = 1/2$.

Another part of Algorithm 4 which is easily amenable to automatic calibration is the MCMC step. We use a random walk Metropolis step, i.e. Algorithm 1 with proposal kernel $\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta}) = \mathrm{N}_p(\boldsymbol{\beta}, \boldsymbol{\Sigma}_{\mathrm{prop}})$, but with $\boldsymbol{\Sigma}_{\mathrm{prop}}$ calibrated to the empirical variance of the particles $\hat{\boldsymbol{\Sigma}}$: $\boldsymbol{\Sigma}_{\mathrm{prop}} = \lambda \hat{\boldsymbol{\Sigma}}$, for some λ. Finally, one may also automatically calibrate the number m of MCMC steps, as in Ridgway (2014), but in our simulations we simply took $m = 3$.

Algorithm 4 tempering SMC

Operations involving index n must be performed for all $n \in 1:N$.

0: Sample $\boldsymbol{\beta}_n \sim q(\boldsymbol{\beta})$ and set $\underline{\delta} \leftarrow 0$.

1: Let, for $\delta \in [\underline{\delta}, 1]$,

$$\mathrm{EF}(\delta) = \frac{1}{N} \frac{\left\{\sum_{n=1}^N w_\gamma(\boldsymbol{\beta}_n)\right\}^2}{\left\{\sum_{n=1}^N w_\gamma(\boldsymbol{\beta}_n)^2\right\}}, \quad u_\delta(\boldsymbol{\beta}) = \left\{\frac{p(\boldsymbol{\beta})p(\mathcal{D}|\boldsymbol{\beta})}{q(\boldsymbol{\beta})}\right\}^\delta.$$

If $\mathrm{EF}(1) \geq \tau$, stop and return $(\boldsymbol{\beta}_n, w_n)_{n=1:N}$ with $w_n = u_1(\boldsymbol{\beta}_n)$; otherwise, use the bisection method (Press et al., 2007, Chap. 9) to solve numerically in δ the equation $\mathrm{EF}(\gamma) = \tau$.

2: Resample according to normalised weights $W_n = w_n / \sum_{m=1}^N w_m$, with $w_n = u_\delta(\boldsymbol{\beta}_n)$; see the supplement for one such resampling algorithm.

3: Update the $\boldsymbol{\beta}_n$'s through m MCMC steps that leaves invariant $\pi_t(\boldsymbol{\beta})$, using e.g. Algorithm 1 with $\kappa(\boldsymbol{\beta}^\star|\boldsymbol{\beta}) = \mathrm{N}_p(\boldsymbol{\beta}, \boldsymbol{\Sigma}_{\mathrm{prop}})$, $\boldsymbol{\Sigma}_{\mathrm{prop}} = \lambda\hat{\boldsymbol{\Sigma}}$, where $\hat{\boldsymbol{\Sigma}}$ is the empirical covariance matrix of the resampled particles.

4: Set $\underline{\delta} \leftarrow \delta$. Go to Step 1.

In the end, one obtains essentially a black-box algorithm. In practice, we shall often observe that, for simple datasets, our SMC algorithm automatically reduces to a single importance sampling step, because the efficiency factor of moving from the initial distribution q to the posterior is high enough. In that case, our SMC sampler performs exactly as standard importance sampling.

Finally, we note that the reweighting step and the MCMC steps of Algorithm 4 are easy to parallelise.

5. Numerical study

The point of this section is to compare numerically the different methods discussed in the previous sections, first on several datasets of standard size (that are representative of previous numerical studies), then in a second time on several bigger datasets.

We focus on the following quantities: the marginal likelihood of the data, $p(\mathcal{D})$, and the p marginal posterior distributions of the regression coefficients β_j. Regarding the latter, we follow Faes et al. (2011) in defining the 'marginal accuracy' of approximation q for component j to be

$$\mathrm{MA}_j = 1 - \frac{1}{2}\int_{-\infty}^{+\infty} |q(\beta_j) - p(\beta_j|\mathcal{D})| \, \mathrm{d}\beta_j.$$

This quantity lies in $[0, 1]$, and is scale-invariant. Since the true marginals $p(\beta_j|\mathcal{D})$ are not available, we will approximate them through a Gibbs sampler run for a very long time. To give some scale to this criterion, assume $q(\beta_j) = \mathrm{N}_1(\beta_j; \mu_1, \sigma^2)$, $p(\beta_j|\mathcal{D}) = \mathrm{N}_1(\beta_j; \mu_2, \sigma^2)$, then MA_j is $2\Phi(-\delta/2) \approx 1 - 0.4 \times \delta$ for $\delta = |\mu_1 - \mu_2|/\sigma$ small enough; e.g. 0.996 for $\delta \approx 0.01$, 0.96 for $\delta \approx 0.1$.

In our results, we will refer to the following four prior/model 'scenarios': Gaussian/probit, Gaussian/logit, Cauchy/probit, Cauchy/logit, where Gaussian and Cauchy refer to the two priors discussed in Section 2.1. All the algorithms have been implemented in C++, using the Armadillo and Boost libraries, and run on a standard desktop computer (except when explicitly stated). Results for NUTS were obtained by running STAN (`http://mc-stan.org/`) version 2.4.0.

5.1. **Datasets of moderate size.** Table 1 lists the 7 datasets considered in this section (obtained from the UCI machine learning repository, except Elections, which

Dataset	$n_{\mathcal{D}}$	p
Pima (Indian diabetes)	532	8
German (credit)	999	25
Heart (Statlog)	270	14
Breast (cancer)	683	10
Liver (Indian Liver patient)	579	11
Plasma (blood screening data)	32	3
Australian (credit)	690	15
Elections	2015	52

TABLE 1. Datasets of moderate size (from UCI repository, except Elections, from web-site of Gelman and Hill (2006)'s book): name (short and long version), number of instances $n_{\mathcal{D}}$, number of covariates p (including an intercept)

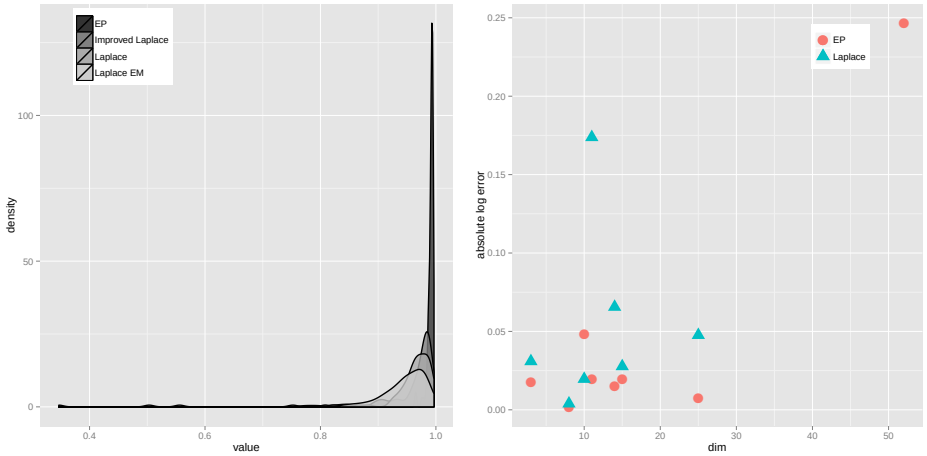
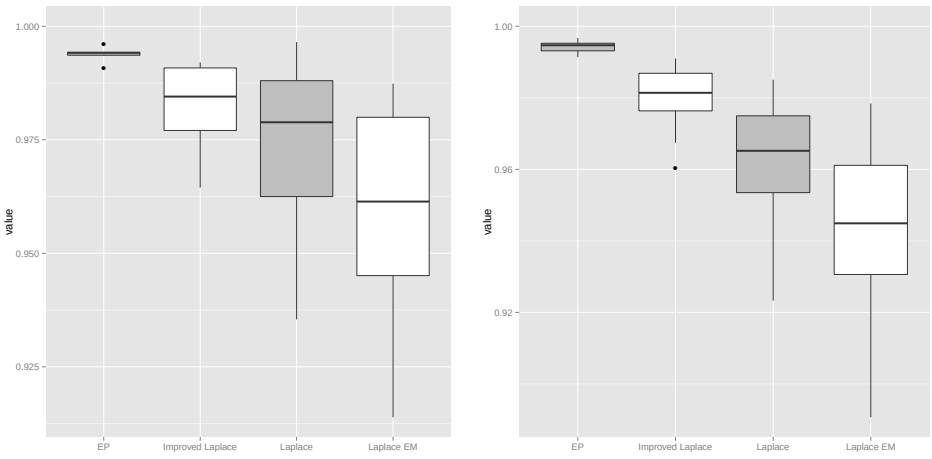


FIGURE 5.1. Comparison of approximation schemes across all datasets of moderate size: marginal accuracies (left), and absolute error for log-evidence versus the dimension p (right); $x-$axis range of the left plot determined by range of marginal accuracies (i.e. marginal accuracy may drop below 0.4 for e.g. Laplace-EM).

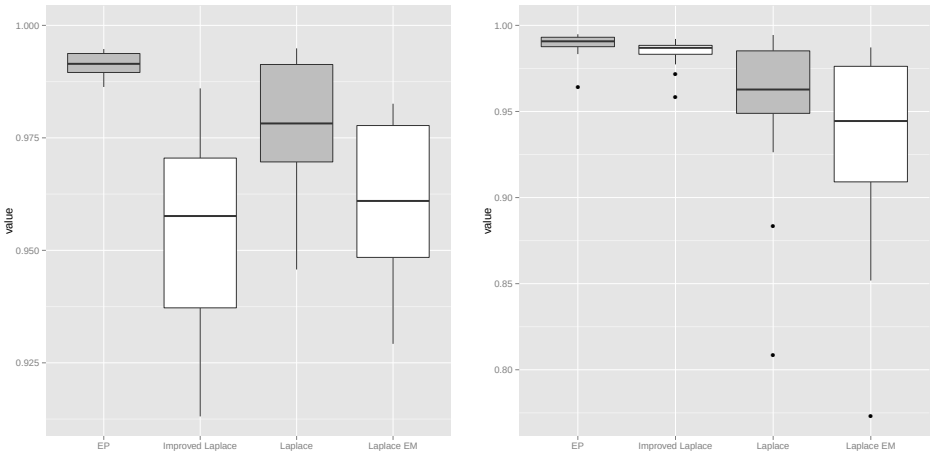
is available on the web page of Gelman and Hill (2006)'s book). These datasets are representative of the numerical studies found in the literature. In fact, it is a superset of the real datasets considered in Girolami and Calderhead (2011), Shahbaba et al. (2011), Holmes and Held (2006) and also (up to one dataset with 5 covariates) Polson et al. (2013). In each case, an intercept have been included; i.e. p is the number of predictors plus one.

5.1.1. *Fast Approximations.* We compare the four approximation schemes described in Section 3: Laplace, Improved Laplace, Laplace EM, and EP. We concentrate on the Cauchy/logit scenario for two reasons: (i) Laplace EM requires a Student prior; and (ii) Cauchy/logit seems the most challenging scenario for EP, as (a) a Cauchy prior is more difficult to deal with than a Gaussian prior in EP ; and (b) contrary to the probit case, the site update requires some approximation; see Section 3.4 for more details.



(A) Pima

(B) Heart



(C) Breast

(D) German

FIGURE 5.2. Box-plots of marginal accuracies across the p dimensions, for the four approximation schemes, and four selected datasets; plots for remaining datasets are in the supplement. For the sake of readability, scale of $y-$axis varies across plots.

Left panel of Fig. 5.1 plots the marginal accuracies of the four approximation schemes across all components and all datasets; Fig. 5.2 does the same, but separately for four selected datasets; results for the remaining datasets are available in the supplement.

EP seems to be the most accurate method on these datasets: marginal accuracy is about 0.99 across all components for EP, while marginal accuracy of the other approximation schemes tend to be lower, and may even drop to quite small values; see e.g. the German dataset, and the left tail in the left panel of Fig. 5.1.

EP also fared well in terms of CPU time: it was at most seven times as intensive as standard Laplace across the considered datasets, and about 10 to 20 times faster than Improved Laplace and Laplace EM. As expected (see Section 3.5). Of course, the usual caveats apply regarding CPU time comparison, and how they may depend on the hardware, the implementation, and so on.

We also note in passing the disappointing performance of Laplace EM, which was supposed to replace standard Laplace when the prior is Student, but which actually performs not as well as standard Laplace on these datasets.

We refer the reader to the supplement for similar results on the three other scenarios, which are consistent with those above. In addition, we also represent the approximation error of EP and Laplace for approximating the log-evidence in the right panel of Fig. 5.1. Again, EP is found to be more accurate than Laplace for most datasets (except for the Breast dataset).

To conclude, it seems that EP may be safely be used as a complete replacement of sampling-based methods on such datasets, as it produces nearly instant results, and the approximation error along all dimensions is essentially negligible.

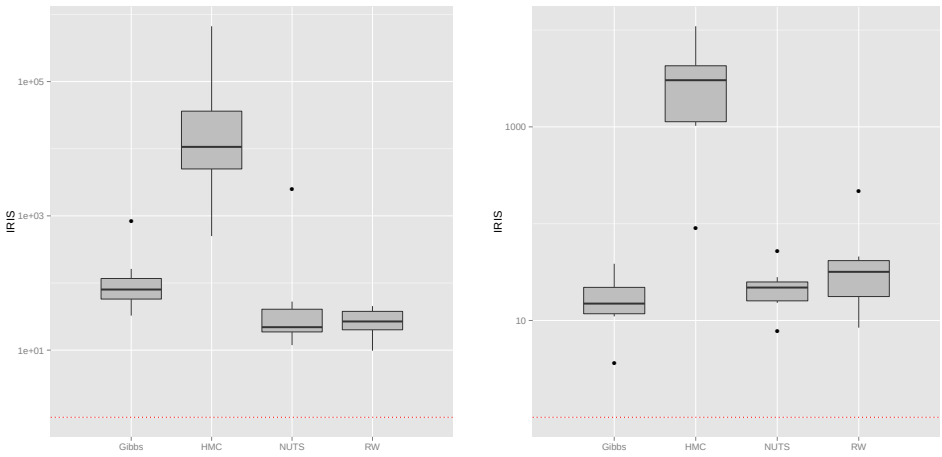
5.1.2. *Importance sampling, QMC.* We now turn to importance sampling (IS), which we deemed our "gold standard" among sampling-based methods, because of its ease of use and other nice properties as discussed in Section 4.1. We use $N = 5 \times 10^5$ samples, and a Gaussian EP proposal. (Results with a Laplace proposal are roughly similar.) We consider first the Gaussian/probit scenario, because this is particularly favorable to Gibbs sampling; see next section. Table 2 reports for each dataset the efficiency factor of IS (as defined in Section 4.1), the CPU time and two other quantities discussed below.

	IS			IS-QMC	
Dataset	EF = ESS/N	CPU time	MT speed-up	MSE improv. (expectation)	MSE improv. (evidence)
Pima	99.5%	37.54 s	4.39	28.9	42.7
German	97.9%	79.65 s	4.51	13.2	8.2
Breast	82.9%	50.91 s	4.45	2.6	6.2
Heart	95.2%	22.34 s	4.53	8.8	9.3
Liver	74.2 %	35.93 s	4.76	7.6	11.3
Plasma	90.0%	2.32 s	4.28	2.2	4.4
Australian	95.6%	53.32 s	4.57	12	20.3
Elections	21.39%	139.48 s	3.87	617.9	3.53

TABLE 2. Performance of importance sampling (IS), and QMC importance sampling (IS-QMC), on all datasets, in Gaussian/probit scenario: efficiency factor (EF), CPU time (in seconds), speed gain when using multi-threading Intel hyper-threaded quad core CPU (Speed gain MT), and efficiency gain of QMC (see text).

We see that all these efficiency factors are all close to one, which means IS works almost as well as IID sampling would on such datasets. Further improvement may be obtained by using either parallelization, or QMC (Quasi-Monte Carlo, see Section 4.2). Table 2 reports the speed-up factor obtained when implementing multi-threading on our desktop computer which has a multi threading quad core CPU (hence 8 virtual cores). We also implemented IS on an Amazon EC2 instance with 32 virtual CPUs, and obtained speed-up factors about 20, and running times below $2s$.

Finally, Table 2 also reports the MSE improvement (i.e. MSE ratio of IS relative to IS-QMC) obtained by using QMC, or more precisely RQMC (randomised QMC), based on a scrambled Sobol' sequence (see e.g. Lemieux, 2009). Specifically, the table reports the median MSE improvement for the p posterior expectations (first column), and the MSE improvement for the evidence (second column). The improvement brought by RQMC varies strongly across datasets.



(A) Median IRIS for the p posterior expectations $\mathbb{E}[\beta_j|\mathcal{D}]$

(B) Median IRIS for the p posterior variances $\mathrm{Var}[\beta_j|\mathcal{D}]$

FIGURE 5.3. IRIS (Inefficiency relative to importance sampling) across all datasets for MCMC schemes and Gaussian/probit scenario; left (resp. right) panel shows median IRIS when estimating the p posterior expectations (resp. the p posterior variances).

The efficiency gains brought by parallelization and QMC may be combined, because the bulk of the computation (as reported by a profiler) is the N likelihood evaluations, which are trivial to parallelize.

It is already clear that other sampling-based methods do not really have a fighting chance on such datasets, but we shall compare them in the next section for the sake of completeness. See also the supplement for results for other scenarios, which are very much in line with those above.

5.1.3. *MCMC schemes.* In order to compare the different sampling-based methods, we define the IRIS (Inefficiency Relative to Importance Sampling) criterion, for a given method M and a given posterior estimate, as follows:

$$\frac{\mathrm{MSE}_M}{MSE_{IS}} \times \frac{\mathrm{CPU}_{IS}}{\mathrm{CPU}_M}$$

where MSE_M (resp. MSE_{IS}) is the mean square error of the posterior estimate obtained from method M (resp. from importance sampling), and CPU_M the CPU time of method M (resp. importance sampling). The comparison is relative to importance sampling *without* parallelisation or quasi-Monte Carlo sampling. In terms of posterior estimates, we consider the expectation and variance of each posterior marginal $p(\beta_j|\mathcal{D})$. We observe that, in both cases, IRIS does not vary much across the p components, so we simply report the median of these p values. Fig 5.3 reports the median IRIS across all datasets. We refer the reader to Section 4.3 for how we tuned these MCMC algorithms.

The first observation is that all these MCMC schemes are significantly less efficient than importance sampling on such datasets. The source of inefficiency seems mostly due to the autocorrelations of the simulated chains (for Gibbs or random walk Metropolis), or, equivalently, the number of leap-frog steps performed at each iteration in HMC and NUTS. See the supplement for ACF's (Autocorrelation plots) to support this statement.

Dataset	$n_{\mathcal{D}}$	p
Musk	476	95
Sonar	208	61
DNA	400	180

TABLE 3. Datasets of larger size (from UCI repository): name, number of instances $n_{\mathcal{D}}$, number of covariates p (including an intercept)

Second, HMC and NUTS do not perform significantly better than random-walk Metropolis. As already discussed, HMC-type algorithms are expected to outperform random walk algorithms as $p \to +\infty$. But the considered datasets seem too small to give evidence to this phenomenon, and should not be considered as reasonable benchmarks for HMC-type algorithms (not to mention again that these algorithms are significantly outperformed by IS on such datasets). We note in passing that it might be possible to get better performance for HMC by finely tuning the quantities ϵ and L on per dataset basis. We have already explained in the introduction why we think this is bad practice, and we also add at this stage that the fact HMC requires so much more effort to obtain good performance (relative to other MCMC samplers) is a clear drawback.

Regarding Gibbs sampling, it seems a bit astonishing that an algorithm specialised to probit regression is not able to perform better than more generic approach on such simple datasets. Recall that the Gaussian/probit case is particularly favourable to Gibbs, as explained in Section 4.3.1. See the supplement for a comparison of MCMC schemes in other scenarios than Gaussian/probit; results are roughly similar, except that Gibbs is more significantly outperformed by other methods, as expected.

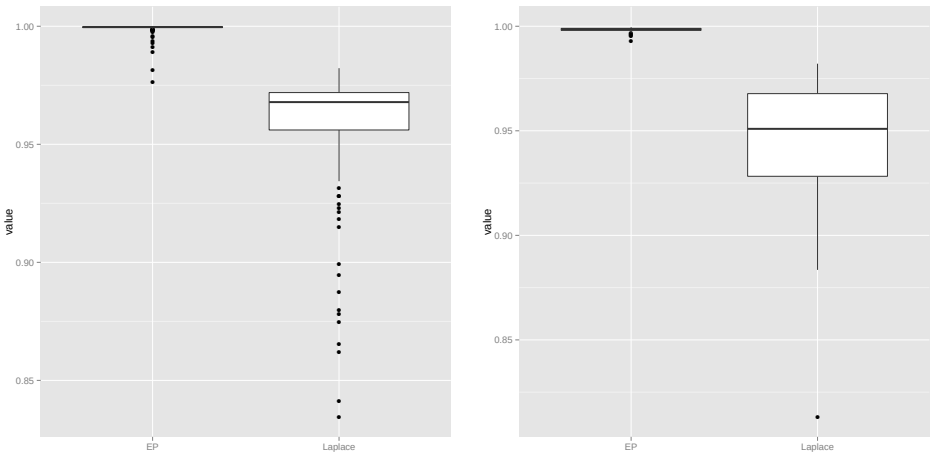
5.2. **Bigger datasets.** Finally, we turn our attention to the bigger datasets summarised by Table 3. These datasets not only have more covariates (than those of the previous section), but also stronger correlations between these covariates (especially Sonar and Musk). We consider the probit/Gaussian scenario.

Regarding fast approximations, we observe again that EP performs very well, and better than Laplace; see Figure 5.4. It is only for DNA (180 covariates) that the EP approximation starts to suffer.

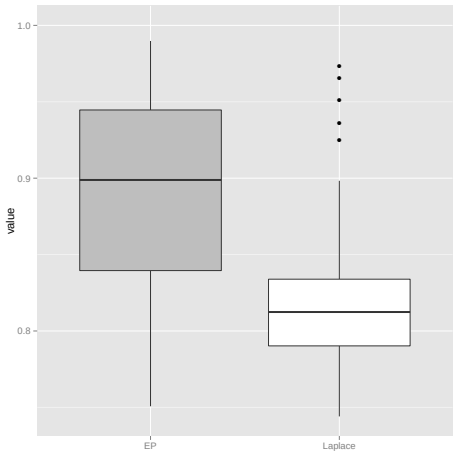
Regarding sampling-based methods, importance sampling may no longer be used as a reference, as the effective sample size collapses to a very small value for these datasets. We replace it by the tempering SMC algorithm described in Section 4.4. Moreover, we did not manage to calibrate HMC so as to obtain reasonable performance in this setting. Thus, among sampling-based algorithms, the four remaining contenders are: Gibbs sampling, NUTS, RWHM (random walk Hastings-Metropolis), and tempering SMC. Recall that the last two are calibrated with the approximation provided by EP.

Figure 5.5 reports the "effective sample size" of the output of these algorithms when run for the same fixed CPU time (corresponding to 5×10^5 iterations of RWHM), for the p posterior expectations (left panels), and the p posterior variances (right panels); here "effective sample size" is simply the posterior variance divided by the MSE of the estimate (across 50 independent runs of the same algorithm).

No algorithm seems to vastly outperform the others consistently across the three datasets. If anything, RWMH seems to show consistently best or second best performance.



(A) Musk (B) Sonar



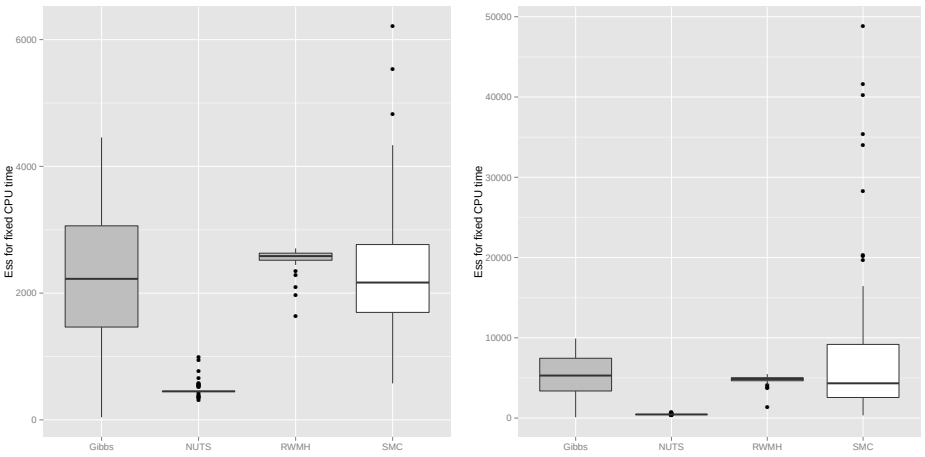
(C) DNA

FIGURE 5.4. Marginal accuracies across the p dimensions of EP and Laplace, for datasets Musk, Sonar and DNA

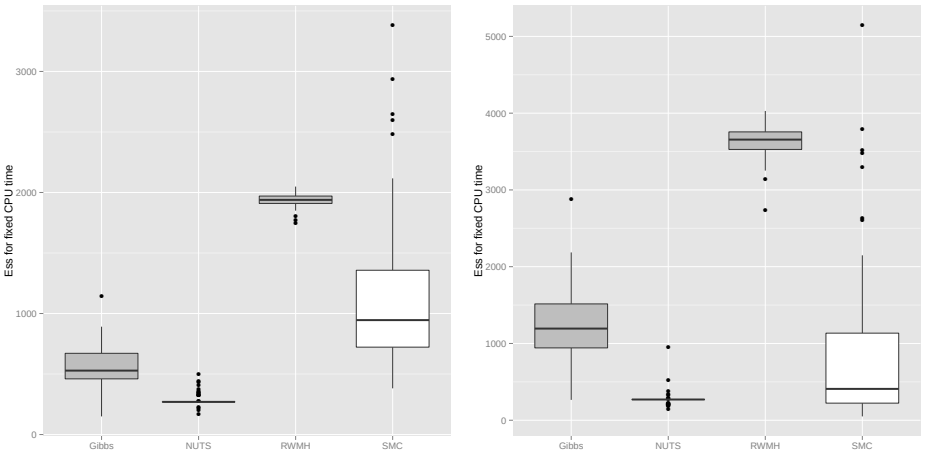
Still, these results offer the following insights. Again, we see that Gibbs sampling, despite being a specialised algorithm, does not outperform significantly more generic algorithms. Recall that the probit/Gaussian scenario is very favourable to Gibbs sampling; in other scenarios (results not shown), Gibbs is strongly dominated by other algorithms.

More surprisingly, RWHM still performs well despite the high dimension. In addition, RHHM seems more robust than SMC to an imperfect calibration; see the DNA example, where the error of the EP approximation is greater.

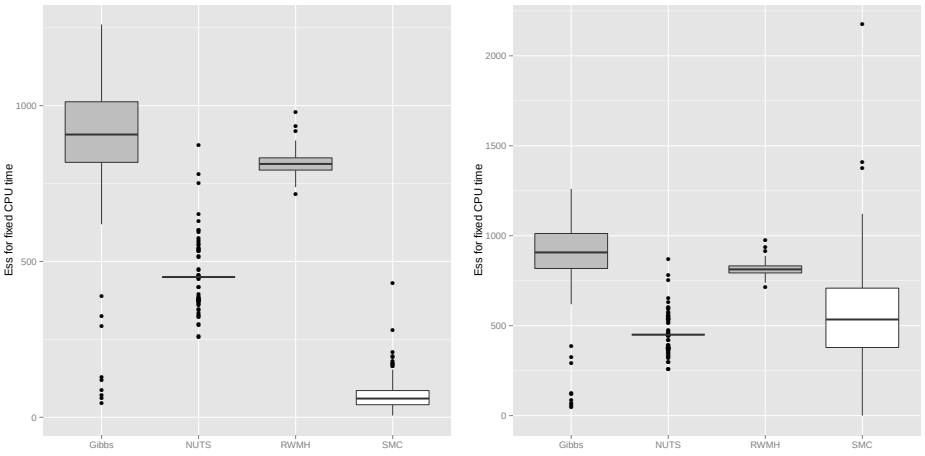
On the other hand, SMC is more amenable to parallelisation, hence on a parallel architecture, SMC would be likely to outperform the other approaches.



(A) Musk



(B) Sonar



(C) DNA

FIGURE 5.5. Effective sample size for a fixed CPU time for sampling-based algorithms: posterior expectations (left), and posterior variances (right) for datasets (from top to bottom): Musk, Sonar, and ADN

6. Variable selection

We discuss in this section the implications of our findings on variable selection. The standard way to formalise variable selection is to introduce as a parameter the binary vector $\boldsymbol{\gamma} \in \{0,1\}^p$, and to define the likelihood

$$p(\mathcal{D}|\boldsymbol{\beta}, \boldsymbol{\gamma}) = \prod_{i=1}^{n_{\mathcal{D}}} F(y_i \boldsymbol{\beta}_{\boldsymbol{\gamma}}^T \boldsymbol{x}_{\boldsymbol{\gamma},i})$$

where $\boldsymbol{\beta}_{\boldsymbol{\gamma}}$ (resp. $\boldsymbol{x}_{\boldsymbol{\gamma},i}$) is the vector of length $|\boldsymbol{\gamma}|$ that one obtains by excluding from $\boldsymbol{\beta}$ (resp. $\boldsymbol{x}_i$) the components j such that $\gamma_j = 0$. Several priors may be considered for this problem (Chipman et al., 2001), but for simplicity, we will take $p(\boldsymbol{\beta}, \boldsymbol{\gamma}) = p(\boldsymbol{\beta})p(\boldsymbol{\gamma})$ where $p(\boldsymbol{\beta})$ is either the Cauchy prior or the Gaussian prior discussed in Section 2.1, and $p(\boldsymbol{\gamma})$ is the uniform distribution with respect to the set $\{0,1\}^p$, $p(\boldsymbol{\gamma}) = 2^{-p}$.

Computationally, variable selection is more challenging than parameter estimation, because the posterior $p(\boldsymbol{\beta}, \boldsymbol{\gamma}|\mathcal{D})$ is a mixture of discrete and continuous components. If p is small, one may simply perform a complete enumeration: for all the 2^p possible values of $\boldsymbol{\gamma}$, approximate $p(\mathcal{D}|\boldsymbol{\gamma})$ using e.g. importance sampling. If p is large, one may adapt the approach of Schäfer and Chopin (2011), as described in the next sections.

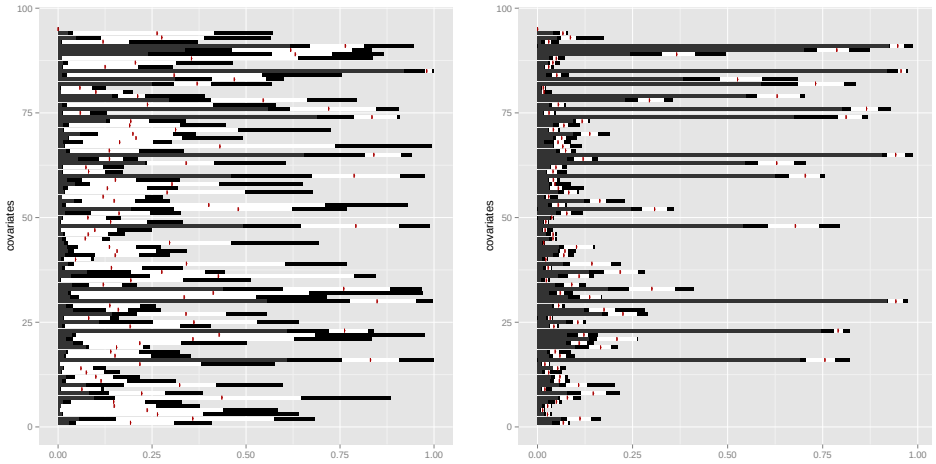
6.1. SMC algorithm of Schäfer and Chopin (2011).

In linear regression, $y_i = \boldsymbol{\beta}_{\boldsymbol{\gamma}}^T \boldsymbol{x}_{\boldsymbol{\gamma},i} + \varepsilon_i$, $\varepsilon_i \sim \mathrm{N}_1(0, \sigma^2)$, the marginal likelihood $p(\mathcal{D}|\boldsymbol{\gamma})$ is available in close form (for a certain class of priors). Schäfer and Chopin (2011) use this property to construct a tempering SMC sampler, which transitions from the prior $p(\boldsymbol{\gamma})$ to the posterior $p(\boldsymbol{\gamma}|\mathcal{D})$, through the tempering sequence $\pi_t(\boldsymbol{\gamma}) \propto p(\boldsymbol{\gamma})p(\mathcal{D}|\boldsymbol{\gamma})^{\delta_t}$, with δ_t growing from 0 to 1. This algorithm has the same structure as Algorithm 4 (with the obvious replacements of the $\boldsymbol{\beta}$'s by $\boldsymbol{\gamma}$'s and so on.) The only difference is the MCMC step used to diversify the particles after resampling. Instead of a random walk step (which would be ill-defined on a discrete space), Schäfer and Chopin (2011) use a Metropolis step based on an independent proposal, constructed from a sequence of nested logistic regressions: proposal for first component γ_1 is Bernoulli, proposal for second component γ_2, conditional on γ_1, corresponds to a logistic regression with γ_1 and an intercept as covariates, and so on. The parameters of these p successive regressions are simply estimated from the current particle system. Schäfer and Chopin (2011) show that their algorithm significantly outperform several MCMC samplers on datasets with more than 100 covariates.

6.2. Adaptation to binary regression.

For binary regression models, $p(\mathcal{D}|\boldsymbol{\gamma})$ is intractable, so the approach of Schäfer and Chopin (2011) cannot be applied directly. On the other hand, we have seen that (a) both Laplace and EP may provide a fast approximation of the evidence $p(\mathcal{D}|\boldsymbol{\gamma})$; and (b) both importance sampling and the tempering SMC algorithm may provide an *unbiased* estimator of $p(\mathcal{D}|\boldsymbol{\gamma})$.

Based on these remarks, Schäfer (2012) in his PhD thesis considered the following extension of the SMC algorithm of Schäfer and Chopin (2011): in the sequence $\pi_t(\boldsymbol{\gamma}) \propto p(\boldsymbol{\gamma})p(\mathcal{D}|\boldsymbol{\gamma})^{\delta_t}$, the intractable quantity $p(\mathcal{D}|\boldsymbol{\gamma})$ is simply replaced by an unbiased estimator (obtained with importance sampling and the Gaussian proposal corresponding to Laplace). The corresponding algorithm remains valid, thanks to pseudo-marginal arguments (see e.g. Andrieu and Roberts, 2009). Specifically, one may re-interpret the resulting algorithm as a SMC algorithm for a sequence of distribution of an extended space, such that marginal in $\boldsymbol{\gamma}$ is exactly the posterior



(A) Gibbs (B) SMC

FIGURE 6.1. Variation of estimated inclusion probabilities $p(\gamma_j = 1|\mathcal{D})$ over 50 runs for the p covariates of Musk dataset: median (red line), 80% confidence interval (white box); the black-box extends until the maximum value.

$p(\mathcal{D}|\boldsymbol{\gamma})$ at time $t = T$. In fact, it may be seen as a particular variant of the SMC^2 algorithm of Chopin et al. (2013).

6.3. **Numerical illustration.** We now compare the proposed SMC approach with the Gibbs sampler of Holmes and Held (2006) for sampling from $p(\boldsymbol{\beta}, \boldsymbol{\gamma}|\mathcal{D})$, on the Musk dataset. Both algorithms were given the same CPU budget (15 minutes), and were run 50 times; see Figure 6.1. Clearly, the SMC sampler provides more reliable estimates of the inclusion probabilities $p(\gamma_j = 1|\mathcal{D})$ on such a big dataset. See also the PhD dissertation of Schäfer (2012) for results consistent with those, on other datasets, and when comparing to the adaptive reversible jump sampler of Lamnisos et al. (2013).

6.4. **Spike and slab.** We also note in passing that a different approach to the variable selection problem is to assign a spike and slab prior to $\boldsymbol{\beta}$ (George and McCulloch, 1993):

$$p(\boldsymbol{\beta}) = \prod_{j=1}^{p} \left\{ \lambda \mathrm{N}_1(\beta_j; 0, v_0^2) + (1-\lambda)\mathrm{N}_1(\beta_j; 0, v_1^2) \right\}, \quad v_0^2 \ll v_1^2$$

where $\lambda \in (0,1)$, v_0^2 and v_1^2 are fixed hyper-parameters. This prior generates a continuous posterior (without point masses at $\beta_j = 0$), which is easier to sample from than the discrete-continuous mixture obtained in the standard formulation of Bayesian variable selection. It would be interesting to see to which extent our discussion and findings extend to this particular type of posteriors; see for instance Hernández-Lobato et al. (2013) for how to deal with such priors in EP.

7. Conclusion and extensions

7.1. **Our main messages to users.** Our first and perhaps most important message to end users is that Bayesian computation (for binary regression) is now sufficiently fast for routine use: if the right approach is used, results may be obtained near instantly on a standard computer, at least on simple datasets.

Concretely, as far as binary regression is concerned, our main recommendation is to always use EP. It is very fast, and its approximation error is negligible in most cases (for such models). EP requires some expertise to implement, but the second author will release shortly a R package that computes the EP approximation for any logit or probit model. The only drawback of EP is the current lack of theoretical support. We learnt however while finishing this manuscript that Simon Barthelmé and Guillaume Dehaene (personal communication) established that the error rate of EP is $\mathcal{O}(n_{\mathcal{D}}^{-2})$ in certain models (where $n_{\mathcal{D}}$ is the sample size). This seems to explain why EP often performs so well.

In case one wishes to assess the EP error, by running in a second step some exact algorithm, we would recommend to use the SMC approach outlined in Section 4.4 (i.e. with initial particles simulated from the EP approximation). Often, this SMC sampler will reduce to a single importance sampling step, and will perform extremely well. Even when it does not, it should provide decent performance, especially if run on (and implemented for) a parallel architecture. Alternatively, on a single-core machine, random walk Metropolis is particularly simple to implement, and performs surprisingly well on high-dimensional data (when properly calibrated using EP).

7.2. **Our main message to Bayesian computation experts.** Our main message to Bayesian computation scientists was already in the title of this paper: leave Pima Indians alone, and more generally, let's all refrain from now on from using datasets and models that are too simple to serve as a reasonable benchmark.

To elaborate, let's distinguish between specialised algorithms and generic algorithms.

For algorithms specialised to a given model and a given prior (i.e. Gibbs samplers), the choice of a "benchmark" reduces to the choice of a dataset. It seems unfortunate that such algorithms are often showcased on small datasets (20 covariates or less), for which simpler, more generic methods perform much better. As a matter of fact, we saw in our simulations that even for bigger datasets Gibbs sampling does not seem to offer better performance than generic methods.

For generic algorithms (Metropolis, HMC, and so on), the choice of a benchmark amounts to the choice of a target distribution. A common practice in papers proposing some novel algorithm for Bayesian computation is to compare that algorithm with a Gibbs sampler on a binary regression posterior for a small dataset. Again, we see from our numerical study that this benchmark is of of limited interest, and may not be more informative than a Gaussian target of the same dimension. If one wishes to stick with binary regression, then datasets with more than 100 covariates should be used, and numerical comparisons should include at least a properly calibrated random walk Metropolis sampler.

7.3. **Big data and the p^3 frontier.** Several recent papers (Wang and Dunson, 2013; Scott et al., 2013; Bardenet et al., 2015) have approached the 'big data' problem in Bayesian computation by focussing on the big $n_{\mathcal{D}}$ (many observations) scenario. In binary regression, and possibly in similar models, the big p problem (many covariates) seems more critical, as the complexity of most the algorithms we have discussed is $\mathcal{O}(n_{\mathcal{D}}p^3)$. Indeed, we do not believe that any of the methods discussed in this paper is practical for $p \gg 1000$. The large p problem may be therefore the current frontier of Bayesian computation for binary regression.

Perhaps one way to address the large p problem is to make stronger approximations; for instance by using EP with an approximation family of sparse Gaussians. Alternatively, one may use a variable selection prior that forbids that the number of active covariates is larger than a certain threshold.

7.4. **Generalising to other models.** We suspect some of our findings may apply more generally to other models (such as certain generalised linear models), but, of course, further study is required to assess this statement.

On the other hand, there are two aspects of our study which we recommend to consider more generally when studying other models: parallelisation, and taking into account the availability of fast approximations. The former has already been discussed. Regarding the latter, binary regression models are certainly not the only models such that some fast approximations may be obtained, whether through Laplace, INLA, Variational Bayes, or EP. And using this approximation to calibrate sampling-based algorithms (Hastings-Metropolis, HMC, SMC, and so on) will often have a dramatic impact on the relative performance of these algorithms. Alternatively, one may also discover in certain cases that these approximations are sufficiently accurate to be used directly.

Acknowledgements

We thank Håvard Rue for insightful comments. The first author is partially funded by Labex ECODEC ANR - 11-LABEX-0047 grant from ANR (Agence Nationale de la Recherche).

References

Albert, J. H. and Chib, S. (1993). Bayesian analysis of binary and polychotomous response data. *J. Am. Statist. Assoc.*, 88(422):669–79.

Andrieu, C. and Roberts, G. (2009). The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725.

Andrieu, C. and Thoms, J. (2008). A tutorial on adaptive MCMC. *Statist. Comput.*, 18(4):343–373.

Bardenet, R., Doucet, A., and Holmes, C. (2015). On Markov chain Monte Carlo methods for tall data. *arXiv preprint arXiv:1505.02827*.

Beskos, A., Pillai, N., Roberts, G., Sanz-Serna, J.-M., and Stuart, A. (2013). Optimal tuning of the hybrid Monte Carlo algorithm. *Bernoulli*, 19(5A):1501–1534.

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer New York.

Chipman, H., George, E. I., and McCulloch, R. E. (2001). *The practical implementation of Bayesian model selection*, pages 65–134.

Chopin, N. (2002). A sequential particle filter for static models. *Biometrika*, 89:539–552.

Chopin, N. (2011). Fast simulation of truncated Gaussian distributions. *Statist. Comput.*, 21(2):275–288.

Chopin, N., Jacob, P., and Papaspiliopoulos, O. (2013). SMC^2: A sequential Monte Carlo algorithm with particle Markov chain Monte Carlo updates. *J. R. Statist. Soc. B*, 75(3):397–426.

Consonni, G. and Marin, J. (2007). Mean-field variational approximate Bayesian inference for latent variable models. *Comput. Stat. Data Anal.*, 52(2):790–798.

Cook, J. D. (2014). Time exchange rate. *The Endeavour (blog)*.

Del Moral, P., Doucet, A., and Jasra, A. (2006). Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. B*, 39:1–38.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Faes, C., Ormerod, J. T., and Wand, M. P. (2011). Variational Bayesian inference for parametric and nonparametric regression with missing data. *Journal of the American Statistical Association*, 106(495):959–971.

Firth, D. (1993). Bias reduction of maximum likelihood estimates. *Biometrika*, 80(1):27–38.

Frühwirth-Schnatter, S. and Frühwirth, R. (2009). Data augmentation and MCMC for Binary and multinomial logit models. In *Statistical Modelling and Regression Structures*, pages 111–132. Physica-Verlag HD.

Gelman, A. and Hill, J. (2006). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Gelman, A., Jakulin, A., Pittau, M. G., and Su, Y.-S. (2008). A weakly informative default prior distribution for logistic and other regression models. *Ann. Appl. Stats.*, 2(4):1360–1383.

George, E. I. and McCulloch, R. E. (1993). Variable Selection via Gibbs Sampling. *J. Am. Statist. Assoc.*, 88(423):881–889.

Girolami, M. and Calderhead, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *J. R. Statist. Soc. B*, 73(2):123–214.

Gramacy, R. B. and Polson, N. G. (2012). Simulation-based regularized logistic regression. *Bayesian Anal.*, 7(3):567–590.

Hernández-Lobato, D., Hernández-Lobato, J. M., and Dupont, P. (2013). Generalized spike-and-slab priors for Bayesian group feature selection using expectation propagation. *J. Mach. Learn. Res.*, 14:1891–1945.

Hoffman, M. and Gelman, A. (2013). The no-U-turn sampler: Adaptively setting path lengths in Hamiltonian monte carlo. *J. Machine Learning Research*, page (in press).

Holmes, C. C. and Held, L. (2006). Bayesian auxiliary variable models for binary and multinomial regression. *Bayesian Anal.*, 1(1):145–168.

Hörmann, W. and Leydold, J. (2005). Quasi importance sampling. Technical report.

Jacob, P., Robert, C. P., and Smith, M. H. (2011). Using Parallel Computation to Improve Independent Metropolis˘Hastings Based Estimation. *J. Comput. Graph. Statist.*, 20(3):616–635.

Jasra, A., Stephens, D., A. Doucet, A., and Tsagaris, T. (2011). Inference for Lévy driven stochastic volatility models via Sequential Monte Carlo. *Scand. J. of Statist.*, 38(1).

Kabán, A. (2007). On Bayesian classification with Laplace priors. *Pattern Recognition Letters*, 28(10):1271–1282.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputation and Bayesian missing data problems. *J. Am. Statist. Assoc.*, 89:278–288.

Lamnisos, D., Griffin, J. E., and Steel, M. F. J. (2013). Adaptive Monte Carlo for Bayesian variable selection in regression models. *J. Comput. Graph. Statist.*, 22(3):729–748.

Lee, A., Yau, C., Giles, M. B., Doucet, A., and Holmes, C. C. (2010). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *J. Comput. Graph. Statist.*, 19(4):769–789.

Lemieux, C. (2009). *Monte Carlo and Quasi-Monte Carlo Sampling (Springer Series in Statistics)*. Springer.

Minka, T. (2001). Expectation Propagation for approximate Bayesian inference. *Proceedings of Uncertainty in Artificial Intelligence*, 17:362–369.

Neal, R. M. (2001). Annealed importance sampling. *Statist. Comput.*, 11:125–139.

Neal, R. M. (2010). MCMC using Hamiltonian dynamics. In Brooks, S., Gelman, A., Jones, G. L., and Meng, X.-L., editors, *Handbook of Markov Chain Monte Carlo*, pages 113–162. Chapman & Hall / CRC Press.

Nickisch, H. and Rasmussen, C. (2008). Approximations for Binary Gaussian Process Classification. *J. Machine Learning Research*, 9(10):2035–2078.

Polson, N. G., Scott, J. G., and Windle, J. (2013). Bayesian inference for logistic models using pólya˘gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.

Ridgway, J. (2014). Computation of Gaussian orthant probabilities in high dimension. *arXiv preprint arXiv:1411.1314*.

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods, 2nd ed.* Springer-Verlag, New York.

Roberts, G. O. and Rosenthal, J. S. (2001). Optimal scaling for various Metropolis-Hastings algorithms. *Statist. Science*, 16(4):351–367.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *J. R. Statist. Soc. B*, 71(2):319–392.

Schäfer, C. (2012). *Monte Carlo methods for sampling high-dimensional binary vectors*. PhD thesis, Université Paris Dauphine.

Schäfer, C. and Chopin, N. (2011). Sequential monte carlo on large binary sampling spaces. *Statistics and Computing*, pages 1–22.

Scott, S. L., Blocker, A. W., and Bonassi, F. V. (2013). Bayes and big data: The consensus monte carlo algorithm. In *Bayes 250*.

Seeger, M. (2005). Expectation Propagation for Exponential Families. Technical report, Univ. California Berkeley.

Shahbaba, B., Lan, S., Johnson, W. O., and Neal, R. M. (2011). Split Hamiltonian Monte Carlo. *Statist. Comput.*, pages 1–11.

Skilling, J. (2006). Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833–860.

Suchard, M. A., Wang, Q., Chan, C., Frelinger, J., Cron, A., and West, M. (2010). Understanding GPU programming for statistical computation: Studies in massively parallel massive mixtures. *J. Comput. Graph. Statist.*, 19(2):419–438.

Tierney, L. and Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *J. Am. Statist. Assoc.*, 81(393):82–86.

Tierney, L., Kass, R. E., and Kadane, J. B. (1989). Fully exponential Laplace approximations to expectations and variances of non-positive functions. *J. Am. Statist. Assoc.*, 84:710–716.

van Gerven, M. A., Cseke, B., de Lange, F. P., and Heskes, T. (2010). Efficient Bayesian multivariate fMRI analysis using a sparsifying spatio-temporal prior. *NeuroImage*, 50(1):150–161.

Wang, X. and Dunson, D. B. (2013). Parallelizing MCMC via Weierstrass sampler. *arXiv preprint arXiv:1312.4605*.