Sandwich Algorithms for Bayesian Variable Selection

Joyee Ghosh[1,*], Aixin Tan[1]

Department of Statistics and Actuarial Science, The University of Iowa, Iowa City, IA 52242, USA.

Abstract

Markov chain Monte Carlo (MCMC) algorithms have greatly facilitated the popularity of Bayesian variable selection and model averaging in problems with high-dimensional covariates where enumeration of the model space is infeasible. A variety of such algorithms have been proposed in the literature for sampling models from the posterior distribution in Bayesian variable selection. Ghosh and Clyde proposed a method to exploit the properties of orthogonal design matrices. Their data augmentation algorithm scales up the computation tremendously compared to traditional Gibbs samplers, and leads to the availability of Rao–Blackwellized estimates of quantities of interest for the original non-orthogonal problem. The algorithm has excellent performance when the correlations among the columns of the design matrix are small, but empirical results suggest that moderate to strong multicollinearity leads to slow mixing. This motivates the need to develop a class of novel sandwich algorithms for Bayesian variable selection that improves upon the algorithm of Ghosh and Clyde. It is proved that the Haar algorithm with the largest group that acts on the space of models is the optimum algorithm, within the parameter expansion data augmentation (PXDA) class of sandwich algorithms. The result provides theoretical insight but using the largest group is computationally prohibitive so two new computationally viable sandwich algorithms are developed, which are inspired by the Haar algorithm, but do not necessarily belong to the class of PXDA algorithms. It is illustrated via simulation studies and real data analysis that several of the sandwich algorithms can offer substantial gains in the presence of multicollinearity.

Keywords: Asymptotic variance, Data augmentation, Markov chain Monte Carlo, Multicollinearity, Rao–Blackwellization.

[*]Corresponding author. Tel.: +1 319 335 0816.

Email addresses: `joyee-ghosh@uiowa.edu` (Joyee Ghosh), `aixin-tan@uiowa.edu` (Aixin Tan)

[1]The authors contributed equally.

[2]Supplementary material includes details of algorithms, statement and proof of a lemma, and theoretical comparisons of sandwich algorithms in some examples.

1. Introduction

Linear regression remains one of the most popular methods for modeling the relationship between a Gaussian response variable and a set of covariates. In recent years this model has received renewed interest for high-dimensional applications where variable selection is commonly employed to identify important covariates. The implementation of Bayesian variable selection in linear regression for general design matrices may prove to be challenging, but the posterior computation can be scaled up tremendously with design matrices having orthogonal columns and certain prior structures. Clyde *et al.* [2] orthogonalized the design matrix in linear regression and developed algorithms that led to better predictive performance, but these cannot be used for variable selection for the original covariates. Ghosh and Clyde [10] proposed a framework that utilized the computational advantages of orthogonal designs but also allowed both model averaging and model selection in terms of the original non-orthogonal covariates. We begin with a brief description of their algorithm and then propose new methods based on sandwich algorithms that use Markov chains with faster convergence rates.

Let $W_o = [w_0, \ldots, w_p]$ denote the $n_o \times (p+1)$ observed design matrix where w_0 is a column of ones corresponding to the intercept, and let $Z_o = (Z_1, \ldots, Z_{n_o})^T$ be the vector of observed response variables. Except for the first column corresponding to the intercept, all other columns of W_o are standardized to have mean zero and norm ${n_o}^{1/2}$. Under the variable selection framework, models can be denoted by $\gamma = (\gamma_1, \ldots, \gamma_p)^T$, where γ_j is binary with $\gamma_j = 1$ when the covariate w_j is included in the model and $\gamma_j = 0$ when w_j is absent from the model. It is assumed that $\beta_j = 0$ when $\gamma_j = 0$, where β_j is the jth regression coefficient and let $p_\gamma = \sum_{j=1}^{p} \gamma_j$. Then the linear model for the observed data under model γ is given by

$$Z_o \mid \beta_\gamma, \phi, \gamma \sim \mathsf{N}(W_{o\gamma}\beta_\gamma, I_{n_o}/\phi), \tag{1}$$

where $W_{o\gamma}$ is the $n_o \times (p_\gamma + 1)$ design matrix, β_γ is the $(p_\gamma + 1)$ dimensional vector of non-zero model specific regression coefficients, and ϕ is the reciprocal of the error variance. The intercept is included in all models and independent prior distributions are assigned as follows:

$$\begin{aligned} p(\beta_0) &\propto 1, \\ p(\phi) &\propto 1/\phi, \\ \beta_j \mid \phi, \gamma_j &\sim \mathsf{N}(0, (\phi\lambda_j)^{-1}\gamma_j) \text{ for } j = 1, \ldots, p, \text{ and} \\ p(\gamma) &= \prod_{j=1}^{p} \pi_j^{\gamma_j}(1-\pi_j)^{1-\gamma_j}, \end{aligned} \tag{2}$$

where λ_j and π_j are fixed hyperparameters.

For a small number of covariates, $p \leq 25$, the posterior probability of any model $p(\gamma \mid Z_o)$ is available in closed form (see Ghosh and Clyde [10] equations (16) and (17) for the exact form). When p is greater than 25, $p(\gamma \mid Z_o)$ is still available up to a normalizing constant, so a standard Gibbs sampler may be constructed that converges to the posterior distribution $p(\gamma \mid Z_o)$. This Gibbs sampler cycles through the p full conditional distributions $p(\gamma_j \mid \gamma_{(j)}, Z_o)$, where $\gamma_{(j)}$ is γ with the jth component removed [9]. For a comprehensive review of Bayesian variable selection see Dellaportas *et al.* [5], Clyde and George [3] and the references therein.

In the data augmentation approach W_o is augmented by a design matrix W_a of dimension $n_a \times (p+1)$, to make the complete design matrix W_c orthogonal. Next Z_o is augmented with a vector of n_a missing response variables, Z_a, and the vector of $n_c = n_o + n_a$ response variables is $Z_c = ({Z_o}^T, {Z_a}^T)^T$. The complete data model is given as

$$Z_c \mid \beta_\gamma, \phi, \gamma, \sim \mathsf{N}(W_{c\gamma}\beta_\gamma, I_{n_c}/\phi). \tag{3}$$

Ghosh and Clyde [10] used a two block Gibbs sampler with invariant distribution $p(\phi, Z_a, \gamma \mid Z_o)$. They sample $(\phi, Z_a) \sim p(\phi, Z_a \mid \gamma, Z_o)$ in one block and $\gamma \sim p(\gamma \mid \phi, Z_a, Z_o)$ in the other block. The first block involves draws from gamma and multivariate normal distributions and the second block involves draws from p independent Bernoulli distributions, all of which are straightforward to sample from. We shall refer to this as the orthogonal data augmentation (ODA) algorithm. More details on the implementation of the ODA algorithm are given in the online supplement for this paper.

Quantities of interest under the original posterior distribution with the non-orthogonal design matrix may be obtained by integrating out the missing data. In practice, this is achieved by averaging over the MCMC iterations. The form of the algorithm naturally leads to the development of Rao–Blackwellized estimates. For example, the estimates of inclusion probabilities are obtained as:

$$\widehat{p}(\gamma_j = 1 \mid Z_o) = K^{-1} \sum_{k=1}^{K} \rho_j({Z_c}^{(k)}, \phi^{(k)}, \lambda_j, \pi_j), \tag{4}$$

where K is the number of iterations after an initial burn-in, and $\rho_j(Z_c, \phi, \lambda_j, \pi_j)$ is the marginal posterior inclusion probability given the missing data, which is available in closed form because of posterior independence. Ghosh and Clyde [10] demonstrated a tremendous computational advantage of their algorithm over the traditional Gibbs sampler.

The ODA algorithm is indeed a data augmentation (DA) algorithm [26], for which we provide a general definition in Section 2.2. It is well-known that DA algorithms can suffer from slow convergence. In the context of linear regression, such a situation may arise when the design matrix exhibits multicollinearity. This may hamper the computational advantage of the ODA algorithm. There is an extensive literature on using the sandwich method [19, 20, 14] to improve the convergence rate of DA algorithms. However, such methods have not been successfully utilized for Bayesian variable selection. This paper introduces novel sandwich algorithms for Bayesian variable selection that employ sandwich moves on the γ space. In Section 2 we review the theory for general sandwich algorithms and show that they dominate the DA algorithm according to two different criteria – the operator norm and the efficiency ordering. In Section 3 we propose a variety of practical sandwich algorithms that improve upon the ODA algorithm. In Sections 4, 5, and 6, we use simulation studies and genuine data analyses to illustrate via empirical studies that sandwich moves can produce substantial gains in the presence of moderate to high multicollinearity. We conclude with some general recommendations in Section 7.

2. Improving DA algorithms

2.1. Ordering reversible Markov chains

In this paper, we restrict our discussion to Markov chains that are reversible. We first introduce the notation. Suppose an MCMC algorithm updates X_n to X_{n+1} using a Markov transition kernel (Mtk) P on the state space X with invariant distribution π_X. Let $L^2(\pi_X) := \{f : \mathsf{X} \to \mathbb{R} \text{ s.t. } \int_{\mathsf{X}} f^2(x)\pi_X(dx) < \infty\}$. Consider the Hilbert space $L_0^2(\pi_X) := \{f \in L^2(\pi_X) \text{ s.t. } \int_{\mathsf{X}} f(x)\pi_X(dx) = 0\}$, equipped with inner product $(f, g) := \int_{\mathsf{X}} f(x)g(x)\pi_X(dx)$, and hence norm $\|f\| = (\int_{\mathsf{X}} f^2(x)\pi_X(dx))^{\frac{1}{2}}$. Then P defines an operator on $L_0^2(\pi_X)$ such that $Pf(x) = \int_{\mathsf{X}} f(x')P(x, dx')$ for any $f \in L_0^2(\pi_X)$. This induces the norm of P by $\|P\| := \sup_{\|f\|=1} \|Pf\|$. Note that P is defined to be an operator on $L_0^2(\pi_X)$, not on $L^2(\pi_X)$, because in the latter case the norm of any Mtk P is 1, hence such an operator norm would not be useful for characterizing the convergence rate of a Markov chain.

In practice, we desire Markov chains with good "mixing" properties. There are two different senses of mixing. On one hand, we prefer Markov chains such that the distribution of X_n approaches the target π_X quickly as n increases. Specifically, a Markov chain is said to be geometrically ergodic, if there exists $\rho < 1$ such that, for any probability measure ν such that $\int \left(\frac{d\nu}{d\pi_X}\right)^2 d\pi_X < \infty$ from which the initial state of the Markov chain is drawn, there exists some $C_\nu < \infty$ that

$$\|\nu P^n(\cdot) - \pi_X(\cdot)\|_{\text{TV}} \leq C_\nu \rho^n \quad \text{for all } n \in N. \tag{5}$$

Here, for a signed measure μ, $\|\mu(\cdot)\|_{\text{TV}} := \sup_{A \subset \mathsf{X}} |\mu(A)|$ stands for its total variation norm, and P^n is the n-step transition kernel. For a reversible Mtk P, Roberts and Rosenthal [24] showed that the corresponding Markov chain is geometrically ergodic if and only if its operator norm $\|P\|$ is strictly less than 1, and $\|P\|$ serves as the smallest possible ρ that satisfies (5) [25, Prop.2]. On the other hand, we would like Monte Carlo estimators of the form $\overline{f}_n = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$ to have small, finite asymptotic variance, denoted by Var(P, f).

The aforementioned mixing properties imply two different ways to order a pair of Mtks, P and Q, that are both reversible with respect to π_X. For an overview, see, for e.g., Mira [21]. First, provided that P is converging with rate $\|P\| < 1$, we say that P is better than Q in the operator norm ordering if $\|P\| \leq \|Q\|$. Further, if $\|P\| < 1$, then Var$(P, f) < \infty$ is guaranteed for any $f \in L^2(\pi_X)$. We say that P is better than Q in the efficiency ordering if Var$(P, f) < \infty$ and Var$(P, f) \leq$ Var(Q, f) for all $f \in L^2(\pi_X)$. This is denoted by $P \geq_{\mathsf{E}} Q$. In the following subsection, we focus on Markov chains that correspond to a popular MCMC method called the DA algorithm, and describe ways to improve upon it subject to the operator norm ordering and the efficiency ordering.

2.2. DA algorithms and a general way to improve them

To study an intractable distribution $\pi_X(x)$ on X, it is sometimes natural from the context of the problem to consider a probability distribution in an augmented space $\mathsf{X}\times\mathsf{Y}$, denoted by $\pi_{X,Y}(x, y)$. Then a DA algorithm consists of the following two steps in each iteration:

1. given x, draw $y \sim \pi_{Y|X}(y|x)$;
2. given y, draw $x' \sim \pi_{X|Y}(x'|y)$.

Note that the Markov chain $\{(X_i, Y_i); i = 1, 2, \ldots\}$ generated by a DA algorithm is not reversible, but the X-chain $\{X_i; i = 1, 2, \ldots\}$ and the Y-chain $\{Y_i; i = 1, 2, \ldots\}$ are reversible Markov chains with respect to π_X and π_Y respectively. In the context of the ODA algorithm we shall assign $X = (\phi, Z_a)$, $Y = \gamma$, and the exact form of $\pi_{X|Y}(x|y)$ and $\pi_{Y|X}(y|x)$ that are needed for sampling are provided in the online supplement. The Rao–Blackwellized estimators of the form (4) depend on the X-chain only, so it is of practical significance to develop theoretical results for this chain, the Mtk of which is

$$P(x, x') = \int_{\mathsf{Y}} \pi_{Y|X}(y|x)\pi_{X|Y}(x'|y)\mu(dy)\,. \tag{6}$$

Here, μ denotes the base measure on Y, such as the counting measure when Y is discrete. The DA algorithm is easy to run when $\pi_{X|Y}$ and $\pi_{Y|X}$ are standard distributions, but like its relative, the EM algorithm, it sometimes suffers from slow convergence.

Below is a general way to improve DA algorithms. For any transition $S(y, y')$ on Y that is reversible with respect to π_Y, do the following in each iteration:

1. given x, draw $y \sim \pi_{Y|X}(y|x)$;
2. given y, draw $y' \sim S(y, y')$;
3. given y', draw $x' \sim \pi_{X|Y}(x'|y')$.

The Mtk that corresponds to the above transition from x to $x'$is given by

$$P_S(x, x') = \int_{\mathsf{Y}} \int_{\mathsf{Y}} \pi_{Y|X}(y|x)S(y, y')\pi_{X|Y}(x'|y')\mu(dy)\mu(dy')\,. \tag{7}$$

Using the terminology in Yu and Meng [27] and Khare and Hobert [16], we call the corresponding algorithm the sandwich algorithm with sandwich move S. It was shown in Hobert and Marchev [14] that any sandwich algorithm is valid in the sense that the reversibility of S with respect to π_Y implies the reversibility of P_S with respect to π_X.

What are some guidelines to find good sandwich algorithms? In order to compare different sandwich algorithms, and to understand their relationship to the original DA algorithm, we first summarize results from Proposition 9 and Theorem 10 of Hobert and Rosenthal [13], and equation 8 of Hobert and Román [15] as the following.

Proposition 1. *Suppose S and S' are two Mtks that are reversible with respect to π_Y. If $S \geq_{\mathsf{E}} S'$, then $P_S \geq_{\mathsf{E}} P_{S'} \geq_{\mathsf{E}} P$ and $\|P_S\| \leq \|P\|$ and $\|P_{S'}\| \leq \|P\|$. Further, if P_S is itself a DA algorithm,* *then* $\|P_S\| \leq \|P_{S'}\|$.

The great advantage of the sandwich idea is that, a reducible Mtk S is not useful by itself to investigate $\pi_Y(\cdot)$, but it can boost the performance of the DA algorithm when these two are combined as in (7) to study $\pi_X(\cdot)$. Further, among those sandwich algorithms with similar computing cost, Proposition 1 allows us to see which ones are more efficient through an easy-to-do comparison of the sandwich moves themselves. We will investigate concrete examples of sandwich algorithms in the Bayesian variable selection context in section 3.

2.3. Improving DA algorithms using Haar algorithms

Perhaps the most well-known subclass of sandwich algorithms in the existing literature is the class of parameter-expansion data augmentation (PXDA) algorithms introduced by Liu and Wu [19]. If there exists a group structure G that acts on Y, then any probability distribution r supported on G would correspond to a PXDA algorithm, and we denote its X-chain operator by P_G^r. It turns out that, with a fixed group G, the entire class of PXDA algorithm $\{P_G^r\}_r$ is dominated by an algorithm called the Haar PXDA, or simply the Haar algorithm, on G. Specifically, let P_G denote the operator of the X-chain of the Haar algorithm, then Hobert and Marchev [14] has shown that $\|P\| \geq \|P_G^r\| \geq \|P_G\|$ and $P \leq_{\mathsf{E}} P_G^r \leq_{\mathsf{E}} P_G$ for any r supported on G. Further, the Haar costs no more than the PXDA per iteration, so we will restrict our attention to Haar algorithms only. A description of PXDA and its relationship to the Haar algorithm can be found in Section 3 of the online supplement.

For reasons given in the beginning of Section 3, we are interested in the case where Y and G are discrete. Let ν_G denote the left Haar measure on G. Then the Haar algorithm can be written as a special sandwich algorithm where the middle step transition $S(y, y')$ in (7) is given by $H_G(y, y') = \sum_{\{g: g\cdot y = y'\}} \pi_Y(g \cdot y)\nu_G(g)$, where $\cdot$ denotes the group action of G on Y. In other words, the step consists of drawing $g \sim f(g) \propto \pi_Y(g \cdot y)\nu_G(g)$, and setting $y' = g \cdot y$. We call the transition from y to y' according to H_G a Haar move.

There are potentially many different choices of G that act on the state space Y. How do the corresponding Haar algorithms compare with each other? The following proposition states that "larger" Gs yield Markov chains with better mixing properties. This new result provides some guidance about the maximum amount of improvement possible with Haar algorithms, based on which we can search for alternative sandwich algorithms that are more cost efficient. The proof of the proposition is in the Appendix.

Proposition 2. *Suppose G and G' are two groups that act on Y. If G' is a subgroup of G, then the Haar algorithm associated with G is better than that with G', and both are better than the original DA algorithm, in the operator norm ordering and the efficiency ordering. In other words, $\|P_G\| \leq \|P_{G'}\| \leq \|P\|$ and $P_G \geq_{\mathsf{E}} P_{G'} \geq_{\mathsf{E}} P$.*

2.4. Sandwich improvements to the DA algorithm that are inspired by the Haar algorithm

Proposition 2 shows that the Haar algorithm with the largest group G that acts on Y, is optimum within the class of PXDA algorithms. However, in practice Haar algorithms that correspond to very large Gs are expensive to run. One practical solution is to resort to smaller groups, which we will discuss in detail in Section 3. Another possibility is to design sandwich algorithms, with moves that approximate the Haar move but are not as costly. Here, we introduce two new kinds of algorithms inspired by the Haar algorithm: the random Haar algorithm and the Metropolis–Hastings (MH) approximation to the Haar algorithm.

First, we propose a random Haar algorithm that combines k groups, $G_1, \cdots, G_k$, that are subgroups of a mother group G. That is, $G_i \subseteq G$ for all i, and each acts on Y and corresponds to a Haar move H_{G_i} on Y. Then for any vector of non-negative weights $(p_1, \cdots, p_k)$ such that $\sum_i p_i = 1$, consider the mixing Mtk $H^* = \sum_i p_i H_{G_i}$. This results in the following reversible Markov chain on X with Mtk $P^* = \sum_i p_i P_{G_i}$. That is, we randomly pick one of the k candidate groups per their importance to act on Y. As part 1 of the following proposition suggests, in terms of performance per iteration, the random Haar algorithm is better than the original DA algorithm, but is inferior to the Haar algorithm based on the mother group G. Part 2 of the proposition compares the convergence rate of P^* and that of the individual P_{G_i}'s. It turns out that no clear-cut comparisons like those in part 1 are available, which is understandable because P^* is in some sense a weighted average of the P_{G_i}'s. Nevertheless, we show that, in terms of operator norm, P^* is guaranteed a convergence rate no worse than that of the least favorable P_{G_i}. Proof of the proposition is in the Appendix.

Proposition 3. *Suppose $G_i \subseteq G$ for all i, and let $G_0 = \cap_{i=1}^k G_i$. Then*

1. $\|P_G\| \le \|P^*\| \le \|P_{G_0}\| \le \|P\|$, *and* $P_G \ge_{\mathsf{E}} P_{G_0} \ge_{\mathsf{E}} P^* \ge_{\mathsf{E}} P$, *and*
2. $\|P^*\| \le \sup_i \|P_{G_i}\|$.

In practice, the random Haar algorithm is potentially useful after taking computing cost into account. The main reason is that it allows the possibility of different groups to act upon Y in different iterations, while keeping the cost per iteration restricted to that of one group.

The second method that we propose is an MH approximation to the Haar move based on a group G. Specifically, suppose the current value is y, then instead of sampling y' from $H_G(y, \cdot)$ directly, we use a uniform proposal on its support $O(y) = \{g \cdot y : g \in G\}$, and accept the proposal with an appropriate MH acceptance probability. We call this a group MH sandwich algorithm based on G. The closer the target marginal distribution of Y is to being uniform over the support, $O(y)$, the better the MH approximation is to the ideal Haar move.

3. Improving the ODA algorithm for Bayesian variable selection

3.1. Applicability of sandwich improvements to the ODA algorithm

The Bayesian variable selection model from Section 1 yields the posterior distribution $p(\phi, Z_a, \gamma \mid Z_o)$. The ODA algorithm is a DA algorithm that updates two blocks of variables, (ϕ, Z_a) and γ in each iteration. Note that all inference of interest can be derived from the posterior of γ, hence a direct application of the sandwich technique would require an extra move on the augmented component (ϕ, Z_a). But this method requires knowledge of the marginal posterior distribution of (ϕ, Z_a) except for a normalizing constant, which involves a summation over 2^p terms and is thus feasible for very small p only. Such a method was implemented by Ghosh and Clyde [10] but it did not lead to improvement in mixing. To overcome this problem, we propose making sandwich moves on γ instead of (ϕ, Z_a). This strategy is feasible for large p because of the following reasons. First, it is possible to design a reversible sandwich move on γ with respect to its marginal posterior distribution. In particular, the sandwich move itself is allowed to be generated from a reducible Markov chain if we desire to focus the computing effort on the most highly correlated covariates. Secondly, samples from (ϕ, Z_a) allow us to obtain Rao–Blackwellized estimators for posterior quantities of the form $E(h(\gamma) \mid Z_o)$ due to the simple form of $p(\gamma \mid \phi, Z_a, Z_o)$.

Using the same notation as in Section 2, let $X = (\phi, Z_a)$, $Y = \gamma$, and $\pi_{X,Y}(x, y) = p(\phi, Z_a, \gamma \mid Z_o)$. Also, let P denote the Mtk (of the X-chain) of the ODA algorithm as in (6), and let P_S denote the Mtk of the sandwich algorithm that employs an extra sandwich move S upon the ODA algorithm as in (7). Here S could correspond to the Haar, random Haar, or any other reversible Mtk for π_Y, which will be further discussed in the following subsections. Since we need reliable estimates based on the MCMC samples, we first make sure that central limit theorems hold for the estimators. Note that the Y-chain of the ODA algorithm lives on a finite state space, so it is uniformly ergodic, hence geometrically ergodic. According to Diaconis *et al.* [6] and Roberts and Rosenthal [23], for DA algorithms, the X-chain shares the same convergence rate as that of the Y-chain. Therefore $\|P\| < 1$, and central limit theorems hold for the estimator based on the ODA algorithm for all functions in $L^2(\pi_X)$. Further, by Proposition 1 we know $\|P_S\| \leq \|P\| < 1$. Hence central limit theorems also hold for estimators based on any sandwich algorithm that improve upon the ODA.

3.2. Haar improvement to the ODA algorithm

The state space Y consists of p-dimensional binary vectors. There are at least two natural ways to move from one binary vector to another using group operations. We first consider a permutation group based move. In the variable selection context, this type of move is very helpful in that it could change a large number of the variables selected in a single step, which is otherwise difficult to achieve. For the set $B = \{1, \cdots, p\}$, the symmetric group, denoted by Sym_B, is the set of all $p!$ permutations equipped with composition as the group operator. We give a few simple examples to illustrate the symbol usage, and refer the readers to Dummit and Foote [7] or any other text on abstract algebra for more background knowledge. Suppose g is a permutation of the set $B = \{1, \cdots, 6\}$ such that $g(1) = 3, g(2) = 1, g(3) = 2, g(4) = 5, g(5) = 4$ and $g(6) = 6$. Then we write $g = (132)(45)$. If we have another permutation $g' = (56)$, then $gg' = (132)(45)(56) = (132)(456)$, where permutations are read from right to left under composition. (The right-to-left rule is conventional, see, for example, Dummit and Foote [7, sec 1.3].) Finally, if $y = (011011)$, then the group action is given by $g \cdot y = (132)(45) \cdot (011011) = (110101)$. We simply write $g \cdot y$ as gy from now on.

A second way to alter a binary vector is through flipping the value of a subset of its components. Let $\text{Flip}_B = \{0, 1\}^p$ denote the group of p-dimension binary vectors that is equipped with component-wise modulo 2 addition as the group operator. For any $g \in \text{Flip}_B$, g acts on y also by component-wise modulo 2 addition. For example, for $p = 6$, the action of flipping the value of the first and the third component of y while leaving all others fixed is $g = (101000)$. Finally, if $y = (011011)$, we have $gy = (101000)(011011) = (110011)$.

To carry out the Haar algorithm based on any discrete group G, note that the left Haar measure on G is always given by $\nu_G(g) = \frac{1}{|G|} I_G(g)$, where $I_G(\cdot)$ stands for the indicator function on G and $|G|$ denotes the cardinality of G. Further, let $O(y) = \{gy : g \in G\} \subseteq \mathsf{Y}$ denote the orbit of y. We show in Lemma 1 in the online supplement that, for any $y \in \mathsf{Y}$ and any $y' \in O(y)$, the number of group elements $g \in G$ that map y to y' is given by $|G|/|O(y)|$. Therefore, the middle step of the Haar algorithm in this context is equivalent to drawing y' from $H_G(y, y') \propto \sum_{\{g:gy=y'\}} \pi_Y(gy)I_G(g) = \frac{|G|}{|O(y)|}\pi_Y(y')I_{O(y)}(y') \propto \pi_Y(y')I_{O(y)}(y')$. That is, the sandwich step amounts to sampling from within the orbit of the current state proportional to the true marginal π_Y. Altogether, the transition function of the Haar algorithm is provided by $P_S(x, x')$ in (7), where $\pi_{X|Y}(x|y)$ and $\pi_{X|Y}(x|y)$ were defined in the ODA algorithm, $S(y, y') = H_G(y, y')$, and μ is the counting measure.

From Proposition 2 in Section 2.3, we know that the above Haar algorithms based on $G = \text{Sym}_B$ and Flip_B both have an iteration-wise advantage over the original DA algorithm. Indeed, the extra move in the Haar algorithm based on Flip_B requires the evaluation of π_Y at all 2^p points in Y, and results in a perfect sampling scheme that is typically prohibitively expensive. The Haar algorithm based on Sym_B involves an extra middle step that requires $p!/\{|y|!(p-|y|)!\}$ evaluations of π_Y, where $|y|$ denotes the number of 1's in the current state y. The evaluations needed for the sandwich move may be done in parallel to reduce computing time, provided that the number of simultaneous evaluations and the number of CPU cores are well coordinated to balance the overhead cost of communication. Alternative to parallel computing, we propose simple ways to specify groups of smaller sizes based on properties of the design matrix.

From our experiments, dimensions of Y that correspond to the highly correlated columns of the design matrix are those that exhibit sticky behavior in the evolution of the Markov chain. They are the direct reason why the DA algorithm has slow mixing rate. Hence, we could expect the biggest gain possible by targeting our effort to alter these dimensions in a sandwich step. Let A denote the collection of indices of all the highly correlated covariates. Then, one could run a Haar algorithm based on the group Sym_A, or that based on the group Flip_A. In case these Haar algorithms still require too many evaluations per move, it is usually possible to further group indices of highly correlated columns into sets $A_1, \cdots, A_k$. (They are often, though not necessarily disjoint.) Then we can form a group G' that is generated by the symmetric groups on the A_is, i.e., $G' =< \text{Sym}_{A_1}, \cdots, \text{Sym}_{A_k} >$, and run the corresponding Haar algorithm. The group G' is a small subgroup of Sym_A, but contains the most important permutations. Suppose the current value is y, and let y_{A_i} denote the subvector of y that corresponds to the set A_i. Then the Haar algorithm based on G' requires only $\Pi_i \left\{|A_i|!/[|y_{A_i}|!(|A_i| - |y_{A_i}|)!]\right\}$ evaluations, which is usually much more affordable compared to the $p!/\{|y|!(p - |y|)!\}$ evaluations needed for the algorithm based on Sym_A.

3.3. Haar inspired and other sandwich improvements to the ODA algorithm

Below we expand our tool box and consider variations and approximations of the Haar improvement to ODA following the general strategies described in Section 2.4. A first variation is the random Haar algorithm. We will apply it to the groups $(G_1, \cdots, G_k) = (\mathrm{Sym}_{A_1}, \cdots, \mathrm{Sym}_{A_k})$. Its vector of weights $(p_1, \cdots, p_k)$ can be either uniform or adjusted to reflect the severity of multicollinearity within the A_is.

A second method is to approximate the Haar algorithm based on a group G with the MH technique. Specifically, given the current value y, one proposes to move to y', that is a draw from the uniform distribution over $O(y) = \{gy : g \in G\}$, and accepts the proposal with an appropriate MH acceptance probability. Such a move is always affordable regardless of the choice of G, as each new move will require only two evaluations of $\pi_Y(\cdot)$, at y and y', in order to calculate the MH acceptance probability.

The MH approximation can be done for any one of the Haar algorithms mentioned earlier, i.e., those based on $G = \mathrm{Sym}_B$, Sym_A, G', Flip_B, or Flip_A. Later, in the simulation and the real data problems, we focus on $G = \mathrm{Sym}_A$ and Flip_A. The reason why we do not do the same for $G = \mathrm{Sym}_B$ or Flip_B is that under these groups, the size of $O(y)$ is often huge, and a uniform proposal over $O(y)$ tends to get very low acceptance rate. Further, we have not implemented an MH approximation for the $G = G'$ case because we are able to afford the exact Haar algorithm based on this group.

Lastly, we consider a random swap move on Y, restricted to the set A. Specifically, this is an

MH move that proposes to swap a randomly chosen pair of 0 and 1 in A of the current y. In other words, one proposes to randomly exchange a covariate in A included in the current model with another one in A that is excluded from the current model, and accept the move with an appropriate probability to maintain the correct invariant distribution π_Y. Note that such a proposal only allows transpositions in A, hence it is more restrictive compared to the MH approximation to the Haar move on A. These exchange based moves are popular in a Bayesian variable selection model that uses the original, non-orthogonal design matrix. Our purpose for adding it is to investigate its performance in conjunction with ODA. In a few very simple cases, it is possible to do theoretical comparisons of the proposed methods. These are presented in an online supplement.

4. Simulations

4.1. High multicollinearity

Our first simulation design is inspired by the multicollinearity example in Section 5.2.2 of George and McCulloch [9]. Here the three pairs of covariates $(1, 2), (3, 4)$ and $(5, 6)$ have correlations higher than 0.995. Moreover almost all the 15 covariates are moderately correlated with each other with correlations around 0.8 in the original example. We preserve this structure for the first 15 covariates and add 85 more noise variables generated as independent $\mathsf{N}(0, 1)$ variables to convert it to a higher dimensional example with $p = 100$. We consider the sample size $n_o = 180$ and generate the response variable exactly as in George and McCulloch [9]. The details are given in the supplement. Following the practice of Ghosh and Clyde [10], we set the hyperparameters of the prior in (2) as $\lambda_j = 1$ and $\pi_j = \frac{1}{2}$ for $j = 1, \cdots, p$, in all examples.

Based on the correlations in the observed design matrix W_o, we decide to perform the sandwich moves on the three pairs of severely correlated variables. We do not consider the traditional Gibbs sampler because it was outperformed by the ODA algorithm by a large margin [10]. Instead, we implement a Metropolis–Hastings algorithm with add/delete steps and random swap proposals which was shown to be more competitive with the ODA algorithm by Ghosh and Clyde [10].

Let $A = \{1, 2, \cdots, 6\}, A_1 = \{1, 2\}, A_2 = \{3, 4\}, A_3 = \{5, 6\}$. We list below the algorithms used in the simulation study:

1. ODA algorithm of Ghosh and Clyde [10],
2. Haar algorithm based on the permutation group $< \text{Sym}_{A_1}, \text{Sym}_{A_2}, \text{Sym}_{A_3} >$,
3. random Haar algorithm which chooses from one of the permutation groups Sym_{A_i} with probability $p_i = 1/3$, for $i = 1, 2, 3$,
4. group MH sandwich algorithm on the permutation group Sym_A,
5. group MH sandwich algorithm on the additive (modulo 2) group Flip_A,
6. random swap sandwich algorithm restricted to $A = \{1, 2, \cdots, 6\}$,
7. Metropolis–Hastings algorithm with add/delete and random swap proposals [4].

To summarize, in the above list the first one is the ODA algorithm that we are trying to improve. Algorithm 2 is the Haar algorithm restricted to a manageable subgroup, which involves a maximum of 8 marginal likelihood evaluations per iteration, in the sandwich step. Algorithms 3-5 are sandwich algorithms that are inspired by Haar moves on algebraic groups, that involve at most two marginal likelihood calculations. Algorithm 6 belongs to the general class of sandwich algorithms, and has similar computing cost as 3-5. All algorithms 2-6 involve these sandwich steps in addition to the sampling steps of ODA. Finally, Algorithm 7 is a traditional Metropolis–Hastings algorithm that does not fall under the class of DA or sandwich algorithms, and involves one marginal likelihood evaluation per iteration.

We initialize all algorithms at the full model, $\gamma = (\gamma_1, \cdots, \gamma_p) = (1, \cdots, 1)$, and run them for one million iterations each to estimate the marginal inclusion probabilities, $p(\gamma_j = 1 \mid Z_o)$. Figure 1 shows pairwise comparisons of these estimated inclusion probabilities for all the algorithms. The diagonal panels of Figure 1 show the estimated inclusion probabilities for each algorithm, plotted in the same order, $j = 1, \ldots, 100$. The plots show very close agreement in the estimates produced by the algorithms, which suggests that the number of iterations was large enough for the chains to converge to their invariant distributions. For further validation of these results, we run all the algorithms again with a different initial value, specifically the vector γ that corresponds to the null model. The resulting estimates are almost indistinguishable from the ones reported here. To compare the performance of the algorithms, we estimate their corresponding asymptotic variances for estimating $p(\gamma_j = 1 \mid Z_o)$, using the mcmcse [8] package in R. For the ODA and the sandwich algorithms Rao–Blackwellized estimates are used as in equation (4), whereas for the Metropolis–Hastings algorithm the usual Monte Carlo estimates are used because no Rao–Blackwellized estimates are available.

	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8	γ_9	γ_{10}
Haar (permutation group)	3.28	14.80	8.61	7.57	17.66	2.41	0.99	0.98	0.99	0.99
random Haar (permutation group)	2.74	6.90	4.55	4.67	7.24	2.24	0.97	0.95	0.94	0.94
permutation group MH sandwich	4.98	8.01	6.32	6.29	8.54	4.23	0.97	0.97	0.92	0.94
additive group MH sandwich	4.85	7.14	5.56	5.51	7.37	4.81	0.98	0.97	0.90	0.95
random swap sandwich	4.38	7.27	5.94	6.24	6.69	3.93	0.97	0.97	0.98	0.97
Metropolis–Hastings	0.17	0.19	0.20	0.19	0.22	0.17	0.24	0.24	0.23	0.24

Table 1: Estimates of relative efficiency of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$ for $j = 1, \ldots, 10$, for the highly correlated simulated data.

	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8	γ_9	γ_{10}
Haar (permutation group)	1.61	7.25	4.22	3.71	8.66	1.18	0.49	0.48	0.49	0.49
random Haar (permutation group)	1.84	4.63	3.05	3.13	4.86	1.50	0.65	0.64	0.63	0.63
permutation group MH sandwich	3.66	5.89	4.65	4.62	6.28	3.11	0.71	0.71	0.68	0.69
additive group MH sandwich	3.49	5.14	4.00	3.96	5.30	3.46	0.71	0.70	0.65	0.68
random swap sandwich	3.06	5.08	4.15	4.36	4.68	2.75	0.68	0.68	0.69	0.68
Metropolis–Hastings	0.61	0.68	0.71	0.68	0.79	0.61	0.86	0.86	0.82	0.86

Table 2: Running time adjusted estimates of relative efficiency of of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$ for $j = 1, \ldots, 10$, for the highly correlated simulated data.

Table 1 shows the estimated relative efficiency of different algorithms with respect to the ODA algorithm. This is calculated as the ratio of the asymptotic variance of ODA with respect to that of the other algorithms; values larger than 1 indicate that the other algorithms lead to a reduction in variance. The first 10 components are displayed due to limitations of space. The sandwich algorithms result in reduction in variance up to 18 times for Haar, and 8.5 times for the other sandwich algorithms. The last row indicates that the Metropolis–Hastings algorithm is less efficient than ODA. Our empirical results suggest that the components of γ that are not directly affected by the sandwich step show a similar asymptotic variance under both ODA and sandwich algorithms. This is also the case for the components that are not displayed.

The results in Table 1 do not take into account running times of different algorithms. Because the running times for algorithms 2-7 are approximately 2.04, 1.49, 1.36, 1.39, 1.43, and 0.28 times that of ODA in this example, we divide the estimated efficiency in rows 1-6 of Table 1 by these time adjustment factors and report the new values in Table 2. Table 2 shows that the sandwich algorithms are substantially more efficient for the sandwiched components even after adjusting for time. For other components ODA is most efficient when time is incorporated. Among the sandwich algorithms, the Haar algorithm shows some of the largest gains, even after incorporating time. The group MH approximations to Haar yield the best general performance, especially the one based on the permutation group.

4.2. Moderate multicollinearity

We now use a simulation design with $n_o = 100$, $p = 50$, and negligible pairwise correlations among most covariates and moderate correlations among just three pairs. The first two covariates are generated from a bivariate normal distribution with means 0, standard deviations 1, and correlation coefficient 0.8. Covariates 3 and 4 are generated in the same way, and covariates 5 and 6 are designed to have a somewhat smaller correlation 0.5. This leads to sample correlation coefficients 0.839, 0.796, and 0.4965 among the three pairs $(1, 2), (3, 4)$, and $(5, 6)$ respectively. The remaining 44 covariates are generated from independent standard normal distributions, so the sample correlation coefficients for all pairs except the first three are smaller than 0.314 in absolute value. The response variable is generated from a multivariate normal distribution with mean $W_o\boldsymbol{\beta}$ and variance I_{n_o}/ϕ, where

$\boldsymbol{\beta} = (1.5, 0.5, 0.5, -0.5, -0.5, 0.4, 0.4, -0.4, -0.4, -0.4, -0.4, -0.4, -0.4, 0, \ldots, 0)'$, and $\phi =$

$1/2.5^2$, resulting in 12 signals (excluding the intercept) and 38 noise variables. The sign and magnitude of the regression coefficients for positively correlated covariates were chosen to be the same. This reflects the belief that these covariates have similar effect on the response variable. We use the same group structures for performing the sandwich moves as in the previous simulation study in Section 4.1. Each algorithm was run for one million iterations, with initial value specified to be the full model. Plots similar to those in Figure 1 (not displayed here due to space limitation) show very good agreement among estimates provided by all the methods. Further, reruns of the algorithms based on different initial values lead to very similar results. Hence we believe that all the algorithms have stabilized after one million iterations, and summarize our results in Tables 3 and 4.

	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8	γ_9	γ_{10}
Haar (permutation group)	10.89	9.17	8.52	10.80	4.63	4.43	1.12	1.36	1.09	1.22
random Haar (permutation group)	3.24	3.22	3.25	3.50	2.24	2.05	1.03	1.24	1.04	1.15
permutation group MH sandwich	3.10	3.03	3.02	3.57	2.05	2.02	1.10	1.28	1.15	1.04
additive group MH sandwich	2.71	2.54	2.69	2.99	1.95	2.04	1.11	1.18	1.08	1.10
random swap sandwich	2.86	2.74	2.81	2.89	2.20	2.07	1.06	1.28	1.07	1.09
Metropolis–Hastings	0.12	0.10	0.08	0.11	0.07	0.08	0.07	0.07	0.06	0.03

Table 3: Estimates of relative efficiency of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$ for $j = 1, \dots, 10$, for the moderately correlated simulated data.

	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8	γ_9	γ_{10}
Haar (permutation group)	3.60	3.04	2.82	3.58	1.53	1.47	0.37	0.45	0.36	0.40
random Haar (permutation group)	1.90	1.89	1.90	2.04	1.31	1.20	0.60	0.72	0.61	0.67
additive group MH sandwich	1.69	1.59	1.68	1.87	1.22	1.28	0.69	0.74	0.68	0.69
permutation group MH sandwich	1.93	1.88	1.88	2.22	1.27	1.25	0.68	0.79	0.72	0.64
random swap sandwich	1.73	1.66	1.70	1.75	1.33	1.26	0.64	0.78	0.65	0.66
Metropolis–Hastings	0.26	0.23	0.18	0.26	0.17	0.18	0.16	0.16	0.13	0.06

Table 4: Running time adjusted estimates of relative efficiency of of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$ for $j = 1, \dots, 10$, for the moderately correlated simulated data.

Table 3 shows the estimated relative efficiency of different algorithms with respect to the ODA algorithm. The Haar algorithm is 11 times more efficient than ODA for some sandwiched components, while the other algorithms can be up to 3.6 times as efficient as ODA. The last row indicates that the Metropolis–Hastings algorithm is less efficient than ODA, about $1/10$ as efficient as ODA for the first 6 components. The components of γ that are not under a direct sandwich move have similar efficiency as ODA. Table 4 shows the running time adjusted relative efficiencies with respect to ODA. Even after adjusting for running time, the Haar algorithm is 3.6 times more efficient than ODA for some components and is more efficient than ODA for all sandwiched components. As earlier, for components which are not under a sandwich move ODA is most efficient when time is incorporated.

Among the sandwich algorithms the Haar algorithm appears to be the best irrespective of whether running time is taken into account or not. This example demonstrates that sandwich algorithms can be useful even when there is moderate linear dependence among only few of the covariates. The gains in efficiency are smaller compared to the previous high multicollinearity example. This is not surprising because in a low or moderate multicollinearity problem the performance of ODA is already quite good to begin with, so there is less scope of further improvement using sandwich moves.

One reviewer raised an important question: how to tune the different algorithms? In our algorithms the main tuning parameter that needs to be specified is the collection of inclusion indicators, i.e. the subset of $\{\gamma_1, \cdots, \gamma_p\}$, on which the sandwich moves are to be performed. For the group based algorithms an associated group structure needs to be selected as well. On one hand, when designing Haar algorithms, our theoretical results show that larger groups are better. Empirical results suggest that small correlations among the covariates (less than 0.5) do not adversely affect the performance of the original ODA algorithm substantially, so one possibility is to perform sandwich moves on inclusion indicators for all covariates that have correlations bigger than some large threshold with one or more covariates. The choice of this threshold is problem specific because a small threshold may lead to a large group for the sandwich step, associated with an increase in computational cost. Our default choices have been choosing a threshold such that the group size is not too large, say no larger than 16. This guarantees that the computational cost per iteration would not be any larger than 16 times that of ODA. In fact, the cost may be much smaller for permutation groups, because many different permutation operations on the vector of binary variables γ, result in the same proposed model γ', and marginal likelihoods need to be evaluated for distinct proposed models only. There are no such savings in computational cost for additive groups, hence we generally prefer using permutation groups over additive groups when building Haar algorithms. On the other hand, when designing Metropolis-Hastings based sandwich algorithms, the computational cost does not increase with larger group sizes, as we always need only two marginal likelihood evaluations at each iteration. However, in this case, it depends on the nature of the posterior distribution whether selecting larger groups for the sandwich step will lead to larger gains. In general, too large a group may lower the acceptance rate and reduce the amount of improvement in mixing. Finally, we recommend a few short pilot runs for each algorithm with varying thresholds following the principles listed above, in order to get a preliminary idea of a reasonable threshold that will balance the group size and the gain per iteration.

5. Protein Data

We consider the protein activity data previously analyzed by Clyde *et al.* [4] as a difficult model selection problem for high correlations among some of the covariates. Here the sample size, $n_o = 96$ and there are $p = 88$ covariates in the full model with all main effects, two-way interaction terms, and quadratic terms for continuous covariates.

We perform sandwich moves on the four pairs of covariates with pairwise correlations greater than 0.995 in absolute value. These are $A_1 = \{1, 18\}$, $A_2 = \{3, 20\}$, $A_3 = \{4, 15\}$, $A_4 = \{13, 17\}$. Also, let $A = \{1, 18, 3, 20, 4, 15, 13, 17\}$. We run all seven algorithms for one million iterations as in the simulation study and report the results in Tables 5 and 6.

The additive group MH sandwich algorithm emerges as a clear winner after adjusting for running time. The permutation group MH sandwich appears to be a close runner up. These algorithms lead to relative efficiency values that are 2-4 times that of ODA, for all components of γ that are

	γ_1	γ_{18}	γ_3	γ_{20}	γ_4	γ_{15}	γ_{13}	γ_{17}
Haar (permutation group)	1.50	1.43	2.45	2.29	1.03	1.02	1.33	1.49
random Haar (permutation group)	1.25	1.30	2.04	1.72	1.03	0.99	1.11	1.25
permutation group MH sandwich	2.43	2.42	3.93	3.35	2.66	2.68	2.50	2.64
additive group MH sandwich	2.77	2.87	3.88	3.42	3.51	3.49	2.63	2.89
random swap sandwich	2.15	2.25	3.27	2.92	2.34	2.27	2.05	2.23
Metropolis–Hastings	0.09	0.09	0.12	0.14	0.08	0.08	0.08	0.10

Table 5: Estimates of relative efficiency of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$, for the components of γ on which sandwich moves were employed, for the protein data.

	γ_1	γ_{18}	γ_3	γ_{20}	γ_4	γ_{15}	γ_{13}	γ_{17}
Haar (permutation group)	0.98	0.93	1.59	1.49	0.67	0.66	0.87	0.97
random Haar (permutation group)	0.91	0.95	1.49	1.26	0.75	0.72	0.81	0.91
permutation group MH sandwich	1.73	1.73	2.81	2.39	1.90	1.91	1.79	1.89
additive group MH sandwich	2.01	2.08	2.81	2.48	2.54	2.53	1.91	2.10
random swap sandwich	1.49	1.56	2.27	2.02	1.62	1.57	1.42	1.55
Metropolis–Hastings	0.31	0.29	0.42	0.47	0.26	0.26	0.28	0.35

Table 6: Running time adjusted estimates of relative efficiency of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$, for the components of γ on which sandwich moves were employed, for the protein data.

under a sandwich move. Even after taking into account time, Table 6 shows that these algorithms can double or triple the efficiency for many components with respect to ODA. For components of γ that are not under a direct sandwich move, the asymptotic variances are similar for ODA and the sandwich algorithms, and so usually the ODA algorithm is the most efficient after adjusting for time. The Metropolis–Hastings algorithm has larger asymptotic variance than all other algorithms as earlier.

6. Biscuit Dough Data

We consider the biscuit dough dataset, which was previously analyzed by Brown *et al.* [1] and more recently by [11]. The dataset was obtained from a near-infrared (NIR) spectroscopy experiment used to analyze the composition of biscuit dough pieces, and it is available as part of the `R` package `ppls` [17]. For each biscuit, the NIR reflectance spectrum is a continuous curve measured at several uniformly spaced wavelengths. The dataset contains measurements at 700 wavelengths which are considered as the covariates. Following the idea of other authors [11] we first thinned the reflectance spectra to 100 evenly placed wavelengths between 1202nm to 2398nm to reduce the model space to a more computationally manageable size of $2^p = 2^{100}$. The measurements from consecutive wavelengths are very highly correlated so the thinning does not result in much loss of information. The response variable is taken to be the percentage of water in each dough. We combined the training and test samples available in the R package resulting in a sample size of $n_o = 70$.

This is a more difficult dataset compared to the protein dataset because the number of covariates is larger than the sample size, and there is very high multicollinearity. The lowest pairwise correlation is around 0.6 and each covariate has 0.995 or higher correlation with at least one other covariate in the dataset. To focus on the most highly correlated covariates for the sandwich step we choose the ones which have correlation 0.9999 or higher with one or more covariates. This leads to choosing the measurements at the 11 wavelengths: $22, 23, 37, 38, 39, 47, 48, 50, 51, 52, 53$. The pairwise correlations among any two of these 11 covariates is 0.986 or higher so we perform sandwich moves on all of them together.
Let $A = \{22, 23, 37, 38, 39, 47, 48, 50, 51, 52, 53\}$, then the algorithms implemented in this case are:

1. ODA algorithm of Ghosh and Clyde [10],
2. group MH sandwich algorithm on the permutation group Sym_A,
3. group MH sandwich algorithm on the additive (modulo 2) group Flip_A,
4. random swap sandwich algorithm restricted to A,
5. Metropolis–Hastings algorithm with add/delete and random swap proposals [4].

We do not implement the Haar algorithm because it would be too expensive to run on such a large group. We run all the above algorithms for one million iterations and report our findings in Tables 7 and 8.

	γ_{22}	γ_{23}	γ_{37}	γ_{38}	γ_{39}	γ_{47}	γ_{48}	γ_{50}	γ_{51}	γ_{52}	γ_{53}
permutation group MH sandwich	8.0	7.5	8.2	8.1	8.3	7.3	7.3	7.1	7.2	7.3	7.5
additive group MH sandwich	27.4	24.1	26.5	28.6	26.3	23.5	24.5	25.1	22.7	22.9	27.0
random swap sandwich	6.0	5.2	5.5	5.6	5.5	5.0	5.0	5.2	4.8	5.1	5.2
Metropolis–Hastings	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Table 7: Estimates of relative efficiency of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$, for the components of γ on which sandwich moves were employed, for the biscuit dough data.

	γ_{22}	γ_{23}	γ_{37}	γ_{38}	γ_{39}	γ_{47}	γ_{48}	γ_{50}	γ_{51}	γ_{52}	γ_{53}
permutation group MH sandwich	5.3	5.0	5.5	5.4	5.5	4.9	4.9	4.7	4.8	4.9	5.0
additive group MH sandwich	19.7	17.3	19.1	20.6	18.9	16.9	17.6	18.1	16.3	16.5	19.4
random swap sandwich	4.3	3.7	3.9	3.9	3.9	3.5	3.5	3.7	3.4	3.6	3.7
Metropolis–Hastings	2.0	1.8	1.9	1.7	1.8	1.7	1.7	1.8	1.8	1.8	1.8

Table 8: Running time adjusted estimates of relative efficiency of different algorithms with respect to the ODA algorithm in estimating $p(\gamma_j = 1 \mid Z_o)$, for the components of γ on which sandwich moves were employed, for the biscuit dough data.

For this analysis, the additive group MH sandwich algorithm emerges as a clear winner before and after adjusting for running time. For same iterations it is at least 22 times as efficient as ODA in all sandwiched components and up to 29 times as efficient as ODA for some. The time adjusted efficiencies are at least 16 times that of ODA and as much as 20 for some components. The other sandwich algorithms all show decent gains after adjusting for time but not as remarkable as this one. This example illustrates that the performance of ODA may be affected the most when there is very high multicollinearity and in these situations sandwich moves can bring a huge improvement over ODA. In this example all algorithms, including the Metropolis–Hastings algorithm, appear to be better than ODA after taking into account time. For components of γ that are not under a direct sandwich move, the efficiencies are similar for ODA and the sandwich algorithms, before adjusting for time. So for these components usually the Metropolis–Hastings algorithm is the most efficient after adjusting for time, it's efficiency being about twice as ODA for all components.

7. Discussion

For Bayesian variable selection problems with moderate to high correlations among some of the covariates, we find that well designed sandwich algorithms can significantly improve the ODA algorithm in estimating functions that exhibit large standard errors due to multicollinearity. Empirical results show that the efficiency of estimators of marginal inclusion probabilities of covariates whose inclusion indicators are under direct sandwich moves can be much higher than that of the ODA algorithm. However, for other covariates that are not under a direct sandwich move, the efficiency of the sandwich algorithms is same as that of the ODA algorithm. This suggests that running the sandwich algorithms for same iterations as ODA will result in better estimates overall. When time is a crucial factor we recommend running both ODA and the sandwich algorithms in parallel and using the appropriate estimates based on their estimated relative efficiency.

Our experience suggests that it is easy to code and automate group MH sandwich algorithms that are inspired by the Haar algorithms, restricted to the set of most highly correlated covariates. They are also not computationally demanding compared to the ODA algorithm, and have promising performances in all the examples that we have tried. Besides, the Haar algorithm based on a smaller permutation subgroup appears to be very competitive for the simulation study, even after adjusting for time. Furthermore, for Haar algorithms, each step has the potential of being implemented in parallel. In the Bayesian variable selection framework, parallel computing has been efficiently used by Hans *et al.* [12] for rapidly exploring large model spaces. It would be interesting to see in the future if parallel computing brings the Haar algorithm more advantage compared to MH approximations.

Acknowledgments

The authors thank the editor, the associate editor, and two reviewers for many useful suggestions that strengthened the paper. The authors also thank Prof. James Hobert for helpful discussions on the topic and constructive suggestions on an earlier draft of the paper. The research of Joyee Ghosh was partially supported by the NSA Young Investigator grant H98230-14-1-0126.

Appendix A: Proof of proposition 2

It is implicitly stated in Liu and Wu [19, Section.5] that the Haar algorithm based on a locally compact group G' is a special PXDA algorithm on G' that corresponds to the left Haar measure $r = \nu_{G'}$. (An alternative proof of the statement in the special case where G' is a discrete group is provided in the online supplement of this paper. Our proof only requires elementary knowledge of discrete groups.) Since $\nu_{G'}$ is supported in $G' \subseteq G$, it is also a probability distribution on G. Hence, $P_{G'}$ can be considered a PXDA algorithm associated with group G. Then P_G is better than $P_{G'}$ in both orderings according to Hobert and Marchev [14].

Appendix B: Proof of proposition 3

Proposition 3 Part (1). First, we study the efficiency ordering of Markov chains by introducing the covariance ordering. Let P and Q be two Mtks, both invariant for π_X. Write $P \leq_1 Q$ if $(h, Ph) \geq (h, Qh)$ for all $h \in L_0^2(\pi_X)$. It was shown in Mira and Geyer [22] that if both P and Q are reversible for π_X, then $P \leq_1 Q$ if and only if $P \leq_E Q$.

Now back to the problem at hand. For each $i = 1, \ldots, k$, since $G_0 \subseteq G_i \subseteq G$, then $P_{G_0} \leq_E P_{G_i} \leq_E P_G$ from Proposition 2. Recall that a Haar algorithm is always reversible with respect to its invariant distribution. This implies that $P_{G_0} \leq_1 P_{G_i} \leq_1 P_G$, i.e., for any $h \in L_0^2(\pi)$, $(h, P_G h) \leq (h, P_{G_i} h) \leq (h, P_{G_0} h)$. Note that $(h, P^* h) = (h, \sum_i p_i P_{G_i} h) = \sum_i p_i (h, P_{G_i} h)$. Then, for any $h \in L_0^2(\pi)$,

$$(h, P_G h) = \sum_i p_i (h, P_G h) \leq (h, P^* h) \leq \sum_i p_i (h, P_{G_0} h) = (h, P_{G_0} h) \,. \tag{8}$$

That is, $P_{G_0} \leq_1 P^* \leq_1 P_G$. Since P_{G_0}, P^*, and P_G are all reversible, it follows that $P_{G_0} \leq_E$

$P^* \leq_E P_G$.

Finally, the above efficiency ordering will imply the operator norm ordering $\|P_G\| \leq \|P^*\| \leq \|P_{G_0}\|$ if we can show that both P_G and P^* are positive operators [13, Theorem 10]. Note that P_G is a Haar algorithm, and hence can be represented as a DA algorithm [14, Theorem 4]. Further, any DA algorithm is positive [18, Lemma 3.2], so P_G is positive. Hence by equation (8), for any $h \in L_0^2(\pi)$, $(h, P^*h) \geq (h, P_G h) \geq 0$. So P^* is also positive. After all, we have shown that $\|P_G\| \leq \|P^*\| \leq \|P_{G_0}\|$.

Proposition 3 Part (2). As shown above, P^* and the P_{G_i}'s are all positive operators. And according to Hobert and Rosenthal [13, Proposition 1(f)], for any positive operator Q, $\|Q\| = \sup_{h \in L_{0,1}^2(\pi_X)}(h, Qh)$, where $L_{0,1}^2(\pi_X) = \{h \in L_0^2(\pi_X) : \|h\| = 1\}$. Now, for all $h \in L_{0,1}^2(\pi_X)$, $(h, P^*h) = \sum_i p_i(h, P_{G_i}h) \leq \sup_i(h, P_{G_i}h)$. Hence

$$\|P^*\| = \sup_h(h, P^*h) \leq \sup_h \sup_i(h, P_{G_i}h) = \sup_i \sup_h(h, P_{G_i}h) = \sup_i \|P_{G_i}\| .$$

So, in terms of operator norm, P^* has a convergence rate no worse than that of the least favorable P_{G_i}.

For completeness, we next discuss the efficiency ordering for P^* and the P_{G_i}'s, and claim that there is no simple way to order them. To see this, consider the special case where $P^* = p_1 P_{G_1} + p_2 P_{G_2}$, where p_1 and p_2 are positive and add up to 1. Let $H = \{h : (h, P_{G_1}h) < (h, P_{G_2}h)\}$ and $H' = \{h : (h, P_{G_1}h) > (h, P_{G_2}h)$. Then as long as neither G_1 nor G_2 is a subgroup of the other group, it is easy to see that both H and H' are non-empty. Then

$$(h, P_{G_1}h) < (h, P^*h) < (h, P_{G_2}h) \text{ for } h \in H$$

and

$$(h, P_{G_2}h) < (h, P^*h) < (h, P_{G_1}h) \text{ for } h \in H'.$$

That is, P^* and the individual operators do not dominate each other in covariance ordering, and hence are incomparable in efficiency ordering.

References

[1] Brown, P. J., Fearn, F., and Vannucci, M. (2001). Bayesian wavelet regression on curves with application to a spectroscopic calibration problem. *Journal of the American Statistical Association* **96**, 454, 398–408.

[2] Clyde, M., DeSimone, H., and Parmigiani, G. (1996). Prediction via orthogonalized model mixing. *Journal of the American Statistical Association* **91**, 1197–1208.

[3] Clyde, M. and George, E. I. (2004). Model uncertainty. *Statistical Science* **19**, 1, 81–94.

[4] Clyde, M. A., Ghosh, J., and Littman, M. L. (2011). Bayesian adaptive sampling for variable selection and model averaging. *Journal of Computational and Graphical Statistics* **20**, 1, 80–101.

[5] Dellaportas, P., Forster, J. J., and Ntzoufras, I. (2002). On Bayesian model and variable selection using MCMC. *Statistics and Computing* **12**, 1, 27–36.

[6] Diaconis, P., Khare, K., and Saloff-Coste, L. (2008). Gibbs sampling, exponential families and orthogonal polynomials (with discussion). *Statistical Science* **23**, 151–200.

[7] Dummit, D. and Foote, R. (2003). *Abstract algebra (Third Edition)*. Wiley.

[8] Flegal, J. M. and Hughes, J. (2012). *mcmcse: Monte Carlo standard errors for MCMC*. R package version 1.0-1.

[9] George, E. I. and McCulloch, R. E. (1997). Approaches for Bayesian variable selection. *Statistica Sinica* **7**, 339–374.

[10] Ghosh, J. and Clyde, M. A. (2011). Rao-Blackwellization for Bayesian variable selection and model averaging in linear and binary regression: A novel data augmentation approach. *Journal of the American Statistical Association* **106**, 495, 1041–1052.

[11] Hans, C. (2011). Elastic net regression modeling with the orthant normal prior. *Journal of the American Statistical Association* **106**, 1383–1393.

[12] Hans, C., Dobra, A., and West, M. (2007). Shotgun stochastic search for "large p" regression. *Journal of the American Statistical Association* **102**, 507–516.

[13] Hobert, J. and Rosenthal, J. (2007). Norm comparisons for data augmentation. *Advances and Applications in Statistics* **7**, 291–302.

[14] Hobert, J. P. and Marchev, D. (2008). A theoretical comparison of the data-augmentation, marginal augmentation and PX-DA algorithms. *Annals of Statistics* **36**, 532–554.

[15] Hobert, J. P. and Román, J. C. (2011). Discussion of "To center or not to center: That is not the question - An ancillarity-sufficiency interweaving strategy (ASIS) for boosting MCMC efficiency" by Y. Yu and X.-L. Meng. *Journal of Computational and Graphical Statistics* **20**, 3, 571–580.

[16] Khare, K. and Hobert, J. (2011). A spectral analytic comparison of trace-class data augmentation algorithms and their sandwich variants. *The Annals of Statistics* **39**, 5, 2585–2606.

[17] Kraemer, N. and Boulesteix, A.-L. (2012). *ppls: Penalized Partial Least Squares*. R package version 1.05.

[18] Liu, J. S., Wong, W. H., and Kong, A. (1994). Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika* **81**, 27–40.

[19] Liu, J. S. and Wu, Y. N. (1999). Parameter expansion for data augmentation. *Journal of the American Statistical Association* **94**, 1264–1274.

[20] Meng, X.-L. and van Dyk, D. A. (1999). Seeking efficient data augmentation schemes via conditional and marginal augmentation. *Biometrika* **86**, 301–320.

[21] Mira, A. (2001). Ordering and improving the performance of Monte Carlo Markov chains. *Statistical Science* **16**, 340–350.

[22] Mira, A. and Geyer, C. J. (1999). Ordering Monte Carlo Markov chains. Tech. Rep. No. 632, School of Statistics, University of Minnesota.

[23] Roberts, G. . and Rosenthal, J. S. (2001). Markov chains and de-initializing processes. *Scandinavian Journal of Statistics* **28**, 489–504.

[24] Roberts, G. O. and Rosenthal, J. S. (1997). Geometric ergodicity and hybrid Markov chains. *Electronic Communications in Probability* **2**, 13–25.

[25] Rosenthal, J. (2003). Asymptotic variance and convergence rates of nearly-periodic Markov chain Monte Carlo algorithms. *Journal of the American Statistical Association* **98**, 461, 169–177.

[26] Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association* **82**, 528–540.

[27] Yu, Y. and Meng, X.-L. (2011). To Center or Not to Center: That Is Not the Question–An Ancillarity-Sufficiency Interweaving Strategy (ASIS) for Boosting MCMC Efficiency. *Journal of Computational and Graphical Statistics* **20**, 531–570.

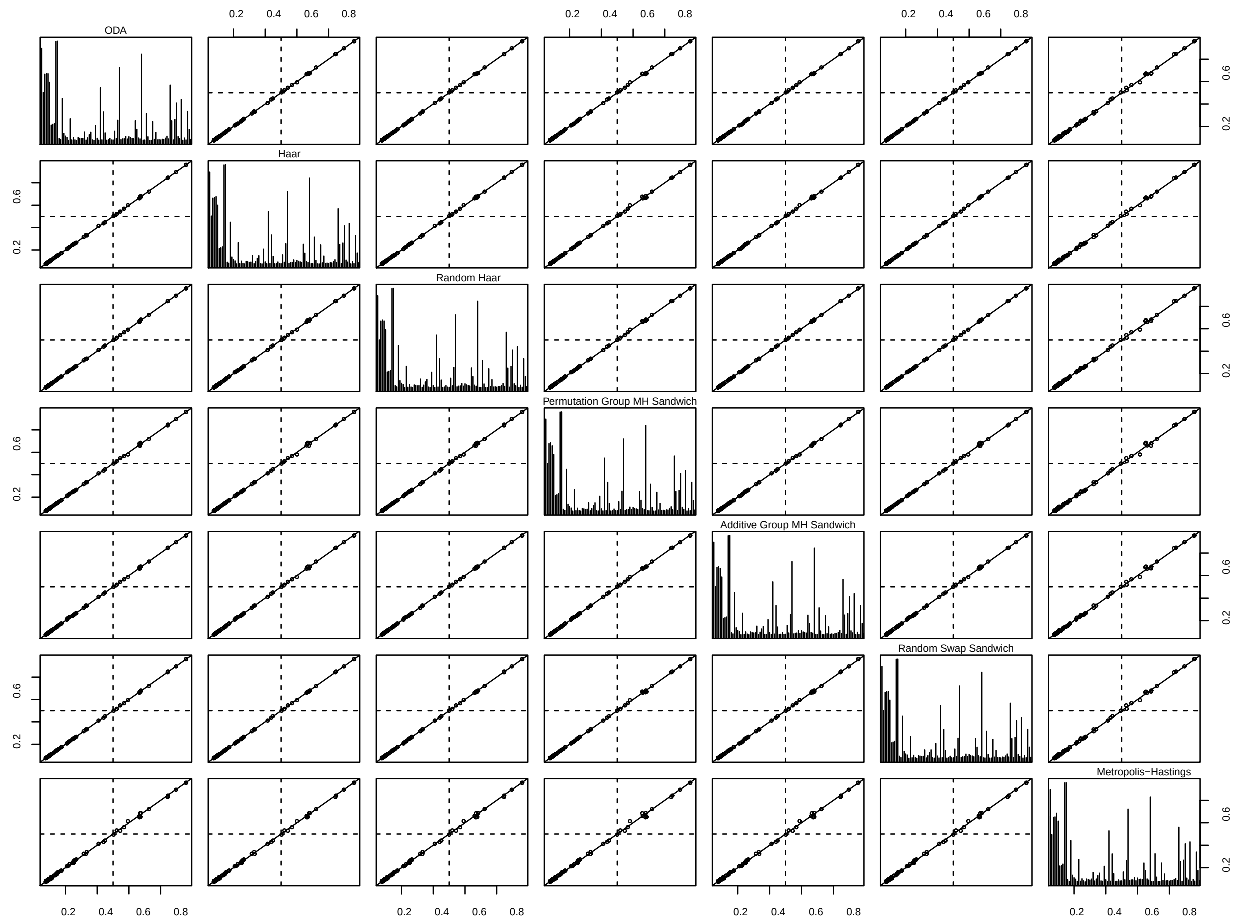

Figure 1: Estimates of marginal posterior inclusion probabilities, $p(\gamma_j = 1 \mid Z_o)$, for $j = 1, \ldots, 100$, based on different algorithms, for the highly correlated simulated data.