blank

Christian Schäfer

Sequential Monte Carlo on large binary sampling spaces

Christian Schäfer * Nicolas Chopin †

January 31, 2011

Abstract

A Monte Carlo algorithm is said to be adaptive if it automatically calibrates its current proposal distribution using past simulations. The choice of the parametric family that defines the set of proposal distributions is critical for a good performance. In this paper, we present such a parametric family for adaptive sampling on high-dimensional binary spaces.

A practical motivation for this problem is variable selection in a linear regression context. We want to sample from a Bayesian posterior distribution on the model space using an appropriate version of Sequential Monte Carlo.

Raw versions of Sequential Monte Carlo are easily implemented using binary vectors with independent components. For high-dimensional problems, however, these simple proposals do not yield satisfactory results. The key to an efficient adaptive algorithm are binary parametric families which take correlations into account, analogously to the multivariate normal distribution on continuous spaces.

We provide a review of models for binary data and make one of them work in the context of Sequential Monte Carlo sampling. Computational studies on real life data with about a hundred covariates suggest that, on difficult instances, our Sequential Monte Carlo approach clearly outperforms standard techniques based on Markov chain exploration by orders of magnitude.

Keywords Adaptive Monte Carlo · Multivariate binary data · Sequential Monte Carlo · Linear regression · Variable selection

1 Introduction

We present a Sequential Monte Carlo (Del Moral et al., 2006) algorithm for adaptive sampling from a binary distribution. A Monte Carlo algorithm is said to be adaptive if it adjusts, sequentially and automatically, its sampling distribution to the problem at hand. Besides Sequential Monte Carlo, important classes of adaptive Monte Carlo are Adaptive Importance Sampling (e.g. Cappé et al., 2008) and Adaptive Markov chain Monte Carlo (e.g. Andrieu and Thoms, 2008).

A central aspect of adaptive algorithms is their need for a parametric family of auxiliary distributions which should have the following three properties: (a) the family is sufficiently flexible to guarantee a reasonable performance in the context of the specific algorithm; (b) it allows to quickly draw independent samples; (c) it can, with reasonable effort, be calibrated using past simulations.

For problems in continuous sampling spaces, the typical example is the multivariate normal distribution, which clearly fulfils (b) and (c), and complies with (a) in many practical problems. In this paper, we propose an analogue for high-dimensional binary sampling spaces.

1.1 Adaptive Monte Carlo on multivariate binary spaces

Our objective is to construct a parametric family for Sequential Monte Carlo on the binary sampling space $\mathbb{B}^d = \{0,1\}^d$, where d is too large to allow for exhaustive enumeration of the whole space $\mathbb{B}^d$. Since there is no multivariate binary family which we can easily parametrise by its first and second order moments like the multivariate normal, the construction of suitable proposal distributions seems more difficult for the discrete adaptive sampling problem than for its continuous counterpart.

The major application for our algorithm is variable selection in linear regression models. In this context, a binary vector $\gamma \in \mathbb{B}^d$ encodes whether each of d possible covariates are included in the linear regression model or not. In a Bayesian framework, and for a judicious choice of prior distributions, we can explicitly calculate the posterior distribution up to a constant.

We want to sample from this distribution in order to approximate quantities like the expected value, that is the marginal probability of inclusion of each variable. Often, the marginal probabilities provide a richer picture of the posterior distribution than a collection of modes found using stochastic optimisation techniques.

*CREST and Université Paris Dauphine · christian.schafer@ensae.fr
†CREST and ENSAE · nicolas.chopin@ensae.fr

1.2 Global versus local methods

Our Sequential Monte Carlo approach to variable selection views a well studied problem from a different angle and provides new perspectives. The reason is two-fold.

Firstly, there is growing evidence that global methods, which track a population of particles, initially well spread over the sampling space $\mathbb{B}^{\mathbf{d}}$, are often more robust than local methods based on Markov chain Monte Carlo. The latter are more prone to get trapped in the neighbourhood of local modes. We largely illustrate this effect in our simulations in Section 6.

Secondly, global methods have the property to be easily parallelisable. Parallel implementations of Monte Carlo algorithms have gained a tremendous interest in the very recent years (Lee et al., 2009; Suchard et al., 2010), due to the increasing availability of multi-core (central or graphical) processing units in standard computers.

1.3 Plan and notations

The paper is organised as follows.

In Section 2, we recapitulate the basics of Bayesian variable selection in linear regression models as the motivating application.

In Section 3, we briefly review the principal Markov chain Monte Carlo methods which are commonly used to integrate with respect to a binary distributions.

In Section 4, we describe an alternative approach to the same problem using Sequential Monte Carlo methods. The key ingredient of this algorithm is a parametric family which is flexible enough to come close to the target distribution.

In Section 5, we extensively discuss approaches for constructing rich parametric families on binary spaces. This is the core of our work. Some of the binary models discussed are not suitable in the framework of our Sequential Monte Carlo algorithm but mentioned for completeness of the survey.

In Section 6, we construct two examples of variable selection problems which yield challenging posterior distributions. We show that standard Markov chain techniques fail to produce reliable estimates of the marginal probabilities while our Sequential Monte Carlo approach successfully copes with the integration problem.

Notation For a vector $\mathbf{x} \in \mathscr{X}^d$, we write $\mathbf{x}_M$ for the sub-vector indexed by $M \subseteq \{1,\dots,d\}$. We write $\mathbf{x}_{i:j}$ if the indices are a complete sequence $i,\dots,j$. We denote by $\mathbf{x}_{-i}$ the sub-vector $\mathbf{x}_{\{1,\dots,d\}\setminus\{i\}}$. We write $|\mathbf{x}|$ for $\sum_{k=1}^d \mathbf{x}_k$.

For a matrix $\mathbf{A}$, the determinant is denoted by $|\mathbf{A}|$. The operator $\operatorname{diag}[\mathbf{x}]$ transforms the vector $\mathbf{x}$ into a diagonal matrix. For a finite set M, we denote by $\#M$ the number of elements in M.

2 Variable selection: A binary sampling problem

The standard linear normal model postulates that the relationship between the observed explained variable $\mathbf{y} \in \mathbb{R}^m$ and the observations $\mathbf{Z} = [\mathbf{z}_1,\dots,\mathbf{z}_d] \in \mathbb{R}^{m,d}$ is

$$\mathbf{y} \mid \beta,\gamma,\sigma^2,\mathbf{Z} \sim \mathscr{N}\left(\mathbf{Z}\operatorname{diag}[\gamma]\beta,\sigma^2\mathbf{I}_m\right).$$

Here, β is a vector of regression coefficients and σ^2 the variance of $\mathbf{y}$. We denote by $\mathbf{I}_m$ the identity matrix and assume the first column $\mathbf{Z}_{\cdot,1}$ to be constant. The parameter $\gamma \in \mathbb{B}^d = \{0,1\}^d$ determines which covariates are included in or dropped from the linear regression model. In total, we can construct 2^d different linear normal models from the data.

We assign a prior distribution $\pi(\beta,\sigma^2,\gamma \mid \mathbf{Z})$ to the parameters. From the posterior distribution

$$\pi(\beta,\sigma^2,\gamma \mid \mathbf{y},\mathbf{Z}) \propto \pi(\mathbf{y} \mid \beta,\sigma^2,\gamma,\mathbf{Z})\,\pi(\beta,\sigma^2,\gamma \mid \mathbf{Z})$$

we may compute the posterior probability of each model

$$\pi(\gamma \mid \mathbf{y},\mathbf{Z}) = \int \pi(\beta,\sigma^2,\gamma \mid \mathbf{y},\mathbf{Z})\,d(\beta,\sigma^2) \tag{1}$$

by integrating out the parameters β and σ^2.

Hierarchical Bayesian model In a purely Bayesian context, we obtain, up to a constant, an explicit formula for the integral in (1) by decomposing the full posterior and choosing conjugate hierarchical priors, that is a normal $\pi(\beta \mid \sigma^2,\gamma,\mathbf{Z})$ and an inverse-gamma $\pi(\sigma^2 \mid \gamma,\mathbf{Z})$. For all Bayesian posterior distributions in this paper, we use the prior distributions

$$\begin{aligned}
&\pi(\beta \mid \sigma,\gamma,\mathbf{Z}) = \mathscr{N}\left(\mathbf{0},\sigma^2 v^2 \operatorname{diag}[\gamma]\right), && \sigma^2 > 0,\\
&\pi(\sigma^2 \mid \gamma,\mathbf{Z}) = \mathscr{I}(w/2,\lambda w/2), && w > 0,\ \lambda > 0,\\
&\pi(\gamma \mid \mathbf{Z}) = \mathscr{U}(\mathbb{B}^d),
\end{aligned}$$

where $\mathscr{I}$ denote an Inverse-Gamma and $\mathscr{U}$ a uniform law.

For our numerical examples in Section 6, we assume not to have any prior information about the data. We follow the recommendations of George and McCulloch (1997) and choose the hyper-parameters

$$w = 4.0, \quad \lambda = \widehat{\sigma}^2_{\mathbf{1}}, \quad v^2 = 10.0/\lambda, \tag{2}$$

where $\widehat{\sigma}^2_{\mathbf{1}}$ is the least square estimate of σ^2 based on the saturated model. The rationale behind this choice is to have a flat prior on β and provide σ^2 with sufficient mass on the interval $(\hat{\sigma}^2_1,\hat{\sigma}^2_0)$, where $\hat{\sigma}^2_0$ denotes the variance of $\mathbf{y}$.

Next, we quickly state the form of the log-posterior mass function. We write $\mathbf{Z}_\gamma$ for $\mathbf{Z}\operatorname{diag}[\gamma]$ without zero columns. Let $\mathbf{b}_\gamma = \mathbf{Z}_\gamma^\intercal\mathbf{y}$ and

$$\mathbf{C}_{\gamma,v}\mathbf{C}_{\gamma,v}^\intercal = \mathbf{Z}_\gamma^\intercal\mathbf{Z}_\gamma + v^{-2}\mathbf{I}_{|\gamma|} \tag{3}$$

a Cholesky decomposition. We denote the least square estimate of σ^2 based on the model γ by

$$\widehat{\sigma}^2_{\gamma,v} = \frac{1}{m}\left(\mathbf{y}^\intercal\mathbf{y} - (\mathbf{C}^{-1}_{\gamma,v}\mathbf{b}_\gamma)^\intercal(\mathbf{C}^{-1}_{\gamma,v}\mathbf{b}_\gamma)\right).$$

We find the log-posterior probability to be

$$\log\pi(\gamma\,|\,\mathbf{y},\mathbf{Z}) = \mu - \textstyle\sum_{i=1}^{|\gamma|}\log c^{(\gamma,v)}_{i,i} - |\gamma|\log(v) - \frac{w+m}{2}\log(w/m + \widehat{\sigma}^2_{\gamma,v}),$$

where μ is an unknown normalization constant.

Bayesian Information Criterion Alternatively, in a Frequentist framework, we choose a model which minimizes a certain criterion. A popular one is the Bayesian Information Criterion introduced by Schwarz (1978), which basically is a second degree Laplace approximation of (1):

$$\log\pi(\gamma\,|\,\mathbf{y},\mathbf{Z}) \approx \mu - \frac{|\gamma|}{2}\log(m) - \frac{m}{2}\log(\widehat{\sigma}^2_\gamma),$$

where $\widehat{\sigma}^2_\gamma = \lim_{v\to\infty}\widehat{\sigma}^2_{\gamma,v}$ is the maximum likelihood estimator of σ^2 based on the model γ. Note that for a large sample size m the Hierarchical Bayesian approach and the Bayesian Information Criterion coincide.

3 Markov chain Monte Carlo on binary spaces

Markov chain Monte Carlo is a well-studied approach to approximate the expected value of a posterior π given by a Bayesian model choice problem (George and McCulloch, 1997). In this section, we rapidly review the standard methods we are going to compare our Sequential Monte Carlo approach against. For background on Markov chain Monte Carlo methods, we refer to standard literature (e.g. Robert and Casella, 2004, chaps. 7-12).

3.1 Framework

The idea is to construct a transition kernel κ, typically some version of a Metropolis-Hastings kernel, which admits π as unique invariant distribution. The distribution of the Markov chain $\mathbf{x}_{t+1} \sim \kappa(\mathbf{x}_t,\cdot)$ started at some randomly chosen point $\mathbf{x}_0 \in \mathbb{B}^d$ converges to π.

We obtain an estimate $\mathbb{E}_\pi(\gamma) \approx n^{-1}\sum_{t=b}^{n+b}\mathbf{x}_t$ of the expected value via the ergodic theorems for Markov chains. The first b states are usually discarded to give the chain some time to converge towards the invariant distribution before we start to average.

Markov chain methods on binary spaces work locally, that is they propose moves to neighbouring models in the Metropolis-Hastings steps. A neighbouring model is a copy of the current model where just a few components are altered.

Algorithm We loop over a uniformly drawn subset of components $I \sim \mathscr{U}(\{M \subseteq \{1,\dots,d\} \mid |M| = k)$ and propose to change the components $i \in I$. The number of components k might be fixed or drawn from some distribution $\mathscr{G}_q$ on the index set $\{1,\dots,d\}$.

Precisely, we take a copy $\mathbf{y}$ of the current state $\mathbf{x}_t$ and replace y_i by $Y_i \sim \mathscr{B}_{p_i(\mathbf{x})}$ for all $i \in I$, where

$$\mathscr{B}_{p_i(\mathbf{x})}(\gamma) = p_i(\mathbf{x})^\gamma(1-p_i(\mathbf{x}))^{1-\gamma}$$

is a Bernoulli distribution with parameter $p_i(\mathbf{x}) \in (0,1)$. We set $\mathbf{x}_{t+1} = y$ with probability

$$\frac{\pi(\mathbf{y})}{\pi(\mathbf{x}_t)}\frac{\prod_{i\in I}\mathscr{B}_{p_i(\mathbf{y})}(\mathbf{x}_t)}{\prod_{i\in I}\mathscr{B}_{p_i(\mathbf{x}_t)}(\mathbf{y})} \wedge 1, \tag{4}$$

and $\mathbf{x}_{t+1} = \mathbf{x}_t$ otherwise. This framework, summarized in Algorithm 1, yields a Markov chain with unique invariant distribution π for any fixed $\mathbf{p} \in (0,1)^d$. The interesting special cases, however, use a $p(x)$ which depends on the current state of the chain.

Algorithm 1 Generic metropolised Gibbs kernel

Input: $\mathbf{x} \in \mathbb{B}^d$
$U \sim \mathscr{U}([0,1])$, $k \sim \mathscr{G}_{k^*}$
$I \sim \mathscr{U}(\{M \subseteq \{1,\dots,d\} \mid |M| = k\})$
$\mathbf{y} \leftarrow \mathbf{x}$
for $i \in I$ **do** $y_i \sim \mathscr{B}_{p_i(\mathbf{x})}$
if $\dfrac{\pi(\mathbf{y})}{\pi(\mathbf{x})}\dfrac{\prod_{i\in I}\mathscr{B}_{p_i(\mathbf{y})}(\mathbf{x})}{\prod_{i\in I}\mathscr{B}_{p_i(\mathbf{x})}(\mathbf{y})} > U$ **then** $\mathbf{x} \leftarrow \mathbf{y}$
return $\mathbf{x}$

Performance We refer to the ratio (4) as the acceptance probability of the Metropolis-Hastings step. In binary spaces, however, accepting a proposal does not imply we are changing the state of the chain, since we are likely to re-propose the current state $y = \mathbf{x}_t$. We are actually interested in how fast the chain explores the state spaces, precisely its mutation probability $\mathbb{P}(\mathbf{x}_{t+1} \neq \mathbf{x}_t)$.

3.2 Standard Markov chain methods

For this section, let $k = 1$ be constant. Algorithm 1 collapses to changing a single component. Instead of independently drawing the index $i \sim \mathscr{U}(\{1,\dots,d\})$, we could also iterate i through a uniformly drawn permutations $\sigma(\{1,\dots,d\})$ of the index set $\{1,\dots,d\}$.

Kernels of this kind are often referred to as metropolised Gibbs samplers, since they proceed component-wise as does the classical Gibbs sampler, but also involve a Metropolis-Hastings step. In the sequel, we discuss some special cases.

Classical Gibbs The Gibbs sampler sequentially draws each component from the full marginal distribution, which corresponds to

$$p_i(\mathbf{x}) \stackrel{def}{=} \pi(\gamma_i = 1 \mid \gamma_{-i} = \mathbf{x}_{-i}) = \frac{\pi(\gamma_i = 1, \gamma_{-i} = \mathbf{x}_{-i})}{\pi(\gamma_i = 1, \gamma_{-i} = \mathbf{x}_{-i}) + \pi(\gamma_i = 0, \gamma_{-i} = \mathbf{x}_{-i})}.$$

By construction, the acceptance probability is 1. The mutation probability is $\pi(\mathbf{y})/(\pi(\mathbf{x}_t) + \pi(\mathbf{y}))$, where $\mathbf{y}$ is a copy of the current state $\mathbf{x}_t$ with component i altered.

Adaptive metropolised Gibbs Nott and Kohn (2005) propose an adaptive version of the metropolised Gibbs. The full marginal distribution $\pi(\gamma_j = 1 \mid \gamma_{-j} = x_{-j})$ is approximated by a linear predictor. In their notation,

$$p_i(\mathbf{x}) \stackrel{def}{=} \left[\left(\psi_i - \frac{\mathbf{W}_{-i}\mathbf{x}_{-i}}{w_{i,i}} \right) \vee \delta \right] \wedge (1 - \delta),$$

where ψ is the estimated mean, $\mathbf{W}^{-1}$ the estimated covariance matrix and $\delta \in (0, 1/2)$ a design parameter which ensures that $p_i(\mathbf{x})$ is a probability. Analogously to our vector notation, $\mathbf{W}_{-i}$ denotes the matrix $\mathbf{W}$ without the ith row and column. We obtain the estimates from the past trajectory of the chain $\mathbf{x}_b, \ldots, \mathbf{x}_{t-1}$ and update them periodically.

The mutation probability is of the same order as that of the Gibbs kernel, but adaption largely avoids the computationally expensive evaluations of π. The non-adaptive Gibbs sampler already requires evaluation of $\pi(\mathbf{y})$ just to produce the proposal $\mathbf{y}$. In contrast, the adaptive metropolised Gibbs samples from a proxy and only evaluates $\pi(\mathbf{y})$ if $\mathbf{y} \neq \mathbf{x}_t$.

Modified metropolised Gibbs Liu (1996) observes that, in comparison to the classical Gibbs kernel, we obtain a more efficient chain from

$$p_i(\mathbf{x}) \stackrel{def}{=} 1 - x_j.$$

Since we always propose to change the current state, the acceptance and mutation probabilities are the same. They amount to $\pi(\mathbf{y})/\pi(\mathbf{x}) \wedge 1$, where $\mathbf{y}$ is a copy of the current state $\mathbf{x}$ with component i altered. Comparing the mutation probabilities of the two kernels, we see that the modified metropolised Gibbs chain moves, on average, faster than the classical Gibbs chain.

3.3 Block updating

The modified metropolised Gibbs easily generalises to the case where k may take values larger than one. Suppose, for example, we propose to change

$$k \sim \mathscr{G}_{k^*}(k) \propto \frac{(1 - 1/k^*)^{k-1}}{k^*} \mathbb{1}_{\{1,\ldots,d\}}(k)$$

components simultaneously, where $\mathscr{G}_{k^*}$ is a truncated geometric distribution. Note that we suggest, on average, to change approximately k^* components. In other words, for larger values of k^*, we are more likely to propose further steps in the sampling space.

Large step proposals improve the mixing properties of the chain and help to escape from the attraction of local modes. They are, however, less likely to be accepted than single component steps which leads to a problem-dependent trade-off. In our numerical examples, we could not observe any benefit from block updating, and we do not further consider it to keep the comparison with our Sequential Monte Carlo method more concise.

3.4 Independent proposals

We can construct a fast mixing Markov chain based on independent proposals. Let q be some distribution with $\pi \ll q$, that is $q(\gamma) = 0 \;\Rightarrow\; \pi(\gamma) = 0$ for all $\gamma \in \mathbb{B}^d$. We propose a new state $y \sim q$ and accept it with probability

$$\frac{\pi(\mathbf{y})}{\pi(\mathbf{x}_t)} \frac{q(\mathbf{x}_t)}{q(\mathbf{y})} \wedge 1. \tag{5}$$

The associated Markov chain has the unique invariant measure π. This kernel is referred to as the independent Metropolis-Hastings kernel, since the proposal distribution is not a function of the current state X_t. The mutation rate is the acceptance rate minus $q(X_t)$, so the two notions practically coincide in large spaces.

Obviously, in order to make this approach work, we need to choose q sufficiently close to π, which implies high acceptance rates on average. In absence of reliable prior information, however, we are not able to produce such a distribution q. We shall, however, use precisely this Markov kernel as part of our Sequential Monte Carlo algorithm. In this context, we can calibrate sequences q_t of proposal distributions to be close to our current particle approximation.

4 Sequential Monte Carlo on binary spaces

In this section, we show how to estimate the expected value with respect to a probability mass function $\pi(\gamma)$ defined on $\mathbb{B}^d$ using Sequential Monte Carlo (Del Moral et al., 2006). This general class of algorithms alternates importance sampling steps, resampling steps and Markov chain transitions, to recursively approximate a sequence of distributions, using a set of weighted 'particles' which represent the current distribution. In the following, we present a version which is tailored to work on binary spaces.

For readers not familiar with Sequential Monte Carlo, the following algorithm described might seem rather complex at first glance. We introduce the steps separately before we look at the complete algorithm. We give comprehensive instructions which correspond exactly to our implementation in order to make our results plausible and easily reproducible for the reader.

4.1 Building a sequence of distributions

The first ingredient of Sequential Monte Carlo is a smooth sequence of distributions $(\pi_t)_{t=0}^{\tau}$, which ends up at the distribution of interest $\pi_\tau = \pi$. The intermediary distributions π_t are purely instrumental: the idea is to depart from a distribution π_0 with broad support and to progress smoothly towards the distribution of interest π.

Initial distribution Theoretically, we can use any π_0 with $\pi \ll \pi_0$ that can sample from as initial distribution. Numerical experiments taught us, however, that premature adjustment of π_0, for example using Markov chain pilot runs, leads to faster but less robust algorithms.

Thus, in practice, we recommend the uniform distribution for its simplicity and reliability. Therefore, in the sequel, we let $\pi_0 = \mathscr{U}(\mathbb{B}^d)$.

Geometric bridge In our context, a natural strategy is the following geometric bridge (Gelman and Meng, 1998; Neal, 2001; Del Moral et al., 2006):

$$\pi_t(\gamma) \overset{def}{\propto} \pi_0(\gamma)^{1-\rho_t}\pi(\gamma)^{\rho_t} \propto \pi(\gamma)^{\rho_t}, \tag{6}$$

where $(\rho_t)_{t=0}^{\tau}$ is an associated real sequence running from zero to one. In the following, we present a procedure to determine an optimal sequence $(\rho_t)_{t=0}^{\tau}$.

4.2 Assigning importance weights

Suppose we have already produced a sample $\mathbf{x}_1^{[t-1]}, \ldots, \mathbf{x}_n^{[t-1]}$ of size n from π_{t-1}. We can roughly approximate π_t by the empirical distribution

$$\pi_t(\gamma) \approx \sum_{k=1}^{n} w_t(\mathbf{x}_k^{[t-1]})\, \delta_{\mathbf{x}_k^{[t-1]}}(\gamma), \tag{7}$$

where the corresponding importance function w_t is

$$w_t(\mathbf{x}_k) \overset{def}{=} \frac{u_t(\mathbf{x}_k)}{\sum_{k=1}^{n} u_t(\mathbf{x}_k)}, \quad u_t(x) \overset{def}{=} \frac{\pi_t(x)}{\pi_{t-1}(x)} = \pi^{\alpha_t}(x). \tag{8}$$

Note that $\alpha_t = \rho_t - \rho_{t-1}$ is the step length at time t. As we choose α_t larger, that is π_t further from π_{t-1}, the weights become more uneven and the accuracy of the importance approximation deteriorates.

Procedure 1 Importance weights

Input: α, π, $\mathbf{X} = (x_1, \ldots, x_n)^\intercal$
  $u_k \leftarrow \pi^\alpha(x_k)$ **for all** $k = 1, \ldots, n$
  $w_k \leftarrow u_k / (\sum_{i=1}^{n} u_i)$ **for all** $k = 1, \ldots, n$
  return $\mathbf{w} = (w_1, \ldots, w_n)$

If we repeat the weighting steps until we reach $\pi_\tau = \pi$, we obtain a classical importance sampling estimate with instrumental distribution π_0. The idea of the Sequential Monte Carlo algorithm, however, is to control the weight degeneracy such that we can intersperse resample and move steps before loosing track of our particle approximation.

Effective sample size We measure the weight degeneracy through the effective sample size criterion, see Kong et al. (1994). In our case, we have

$$\eta(\alpha, \mathbf{x}) \overset{def}{=} \frac{(\sum_{k=1}^{n} w_\alpha(x_k))^2}{n \sum_{k=1}^{n} w_\alpha(x_k)^2} = \frac{(\sum_{k=1}^{n} \pi^\alpha(x_k))^2}{n \sum_{k=1}^{n} \pi^\alpha(x_k)^2} \in [1/n, 1].$$

The effective sample size is 1 if all weights are equal and $1/n$ if all mass is concentrated in a single particle.

For a geometric bridge (6), the effective sample size is merely a function of α. We can thus control the weight degeneracy by judicious choice of the step lengths α_t.

4.3 Finding the step length

We pick a step length α such that the efficient sample size $\eta(\alpha)$ equals a fixed value η^*. Since η is continuous and monotonously increasing in α, we can solve

$$\eta(\alpha, x) = \eta^* \tag{9}$$

using bi-sectional search, see Procedure 2. This approach is numerically more stable than a Newton-Raphson iteration, for the derivative $\partial\eta(\alpha, \mathbf{x})/\partial\alpha$ involves fractions of sums of exponentials which are difficult to handle.

Let α^* be the unique solution to (9). We obtain an associated sequence setting $\rho_t = 1 \wedge (\rho_{t-1} + \alpha^*)$. Thus, the number of steps τ depends on the complexity of the integration problem at hand and is not known in advance.

In other words, for fixed η^*, the associated sequence $(\rho_t)_t^{\tau}$ is a self-tuning parameter. In our simulations, we always choose $\eta^* = 0.9$, which yields convincing results on both example problems in Section 6. Smaller values significantly speed up the Sequential Monte Carlo algorithm but lead to a higher variation in the results.

Procedure 2 Find step length

Input: ρ, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\intercal$
  $l \leftarrow 0$, $u \leftarrow 1.05 - \rho$, $\alpha \leftarrow 0.05$
  repeat
    if $\eta(\alpha, \mathbf{x}) < \eta^*$ **then** $u \leftarrow \alpha$, $\alpha \leftarrow (\alpha + l)/2$
    else $l \leftarrow \alpha$, $\alpha \leftarrow (\alpha + u)/2$
  until $|u - l| < \varepsilon$ **or** $l > 1 - \rho$
  return $\alpha \wedge (1 - \rho)$

4.4 Resampling the system

Suppose we have a sample $\mathbf{X}^{[t-1]} = (\mathbf{x}_1^{[t-1]}, \ldots, \mathbf{x}_n^{[t-1]})$ of size n from π_{t-1} with importance weights as defined in (8). We can obtain a sample $\widehat{\mathbf{X}}^{[t]} = (\hat{\mathbf{x}}_1^{[t]}, \ldots, \hat{\mathbf{x}}_n^{[t]})$ which is approximately

distributed according to π_t by drawing from the empirical approximation defined in (7).

For the implementation of the resampling step, there exist several recipes. We could apply a multinomial resampling (Gordon et al., 1993) which is straightforward. There are, however, more efficient ways like residual (Liu and Chen, 1998), stratified (Kitagawa, 1996) and systematic resampling (Carpenter et al., 1999). We use the latest in our simulations, see Procedure 3.

In the resulting unweighted particle approximation $\widehat{\mathbf{X}}^{[t]}$, the particles with small weights have vanished while the particles with large weights have bee multiplied.

If we repeat the weighting and resampling steps several times, we will rapidly deplete our particle reservoir reducing the number of different particles to a very few. Thus, the particle approximation will be totally inaccurate. The key to fighting the decay of our approximation is the following move step.

Procedure 3 Resample (systematic)

Input: $\mathbf{w} = (w_1, \dots, w_n)$, $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\intercal$
  $v \leftarrow nw,\ j \leftarrow 1,\ c \leftarrow v_1$
  sample $u \sim \mathscr{U}([0,1])$
  for $k = 1, \dots, n$ **do**
    while $c < u$ **do**
      $j \leftarrow j+1,\ c \leftarrow c + v_j$
    end while
    $\hat{\mathbf{x}}_k \leftarrow \mathbf{x}_j,\ u \leftarrow u+1$
  end for
  return $\widehat{\mathbf{X}} = (\hat{\mathbf{x}}_1 \dots, \hat{\mathbf{x}}_n)^\intercal$

4.5 Moving the system

The resampling step leaves us with an unweighted particle approximation $\widehat{\mathbf{X}}^{[t]} = (\hat{\mathbf{x}}_1^{[t]}, \dots, \hat{\mathbf{x}}_n^{[t]})$ of π_t containing multiple copies of many particles. The central idea of the Sequential Monte Carlo algorithm is to diversify the resampled system, replacing the particles by draws from a Markov kernel κ_t with invariant measure π_t.

Since the particle $\mathbf{x}_k^{[0]}$ is, approximately, distributed according to π_t, a draw $\mathbf{x}_k^{[1]} \sim \kappa_t(\mathbf{x}_k^{[0]}, \cdot)$ is again, approximately, distributed according to π_t. We can repeat this procedure over and over without changing the target of the particle approximation.

Note that, even if the particles $\mathbf{x}_k^{[0]} = \dots = \mathbf{x}_m^{[0]}$ are equal after resampling, the particles $\mathbf{x}_k^{[s]}, \dots, \mathbf{x}_m^{[s]}$ are almost independent after sufficiently many move steps. In order to make the algorithm practical, however, we need a transition kernel which is rapidly mixing and therefore diversifies the particle system within a few steps. Therefore, the locally operating Markov kernels reviewed in Section 3 are not suitable. In fact, our numerical experiments suggest that making a Sequential Monte Carlo algorithm work with local kernels is practically impossible.

Therefore, we use a Metropolis-Hastings kernel with independent proposals as described in Section 3.4. Precisely, we construct a kernel κ_t employing a parametric family q_θ on $\mathbb{B}^d$ which, for some θ, is sufficiently close to π_t to allow for high acceptance probabilities.

For this purpose, we fit a parameter θ_t to the particle approximation $(\mathbf{w}_t, \mathbf{X}_t)$ of π_t according to some convenient criterion. The choice of the parametric family q_θ is crucial to a successful implementation of the Sequential Monte Carlo algorithm. We come back to this issue in Section 5.

Particle diversity We need to determine how often we want to move the particle system before we return to the weight-resample step. An easy criterion for the health of the particle approximation $\mathbf{X} = (x_1, \dots, x_n)$ is its particle diversity

$$\zeta(\mathbf{X}) \stackrel{def}{=} \frac{\#\{\mathbf{x}_k \mid k = 1, \dots, n}{n} \in [1/n, 1], \tag{10}$$

that is the proportion of distinct particles. Note that the particle diversity is a quality criterion which has no simple analogue in continuous sampling spaces.

For optimal results, we recommend to keep on moving the particle system until the particle diversity cannot be augmented any longer. In the first steps of the algorithm, π_t is still close to the uniform distribution, and we manage to raise the particle diversity up to one.

As π_t is approaching a strongly multi-modal target distribution π, however, the particle diversity reaches a steady-state we cannot push it beyond. Clearly, even if we could draw a particle system independently from π, the particle diversity would be a lot smaller than one, since we would draw the modes of π several times.

Procedure 4 Move

Input: $\mathbf{X}^{[0]} = (\mathbf{x}_1^{[0]}, \dots, \mathbf{x}_n^{[0]}) \sim \pi_t$
  $\kappa(\mathbf{y}, \gamma)$ such that $\pi_t(\gamma) = \sum_{y \in \mathbb{B}^d} \pi_t(\mathbf{y}) \kappa(\mathbf{y}, \gamma)$
  $s \leftarrow 1$
  repeat
    sample $\mathbf{x}_k^{[s]} \sim \kappa(\mathbf{x}_k^{[s-1]}, \cdot)$ **for all** $k = 1, \dots, n$
  until $|\zeta(\mathbf{X}^{[s]}) - \zeta(\mathbf{X}^{[s-1]})| < 0.02$ **or** $\zeta(\mathbf{X}^{[s]}) > 0.95$
  return $\mathbf{X}^{[s]} = (\mathbf{x}_1^{[s]} \dots, \mathbf{x}_n^{[s]})^\intercal$

4.6 The Resample-move algorithm

Finally, we summarize the complete Sequential Monte Carlo method in Algorithm 2. Note that, in practice, the sequence $\pi_t = \pi^{\rho_t}$ is not indexed by t but rather by ρ_t, that is the counter t is only given implicitly.

For an efficient implementation, we recommend to store the values $\pi(\mathbf{x}_1), \dots, \pi(\mathbf{x}_n)$ and $q_\theta(\mathbf{x}_1), \dots, q_\theta(\mathbf{x}_n)$ to avoid unnecessary evaluations. When updating the latter set, we

Algorithm 2 Resample-move

Input: $\pi\colon \mathbb{B}^d \to [0,\infty)$
  sample $\mathbf{x}_k \overset{\text{iid}}{\sim} \mathscr{U}(\mathbb{B}^d)$ **for all** $k = 1,\dots,n$.
    $\alpha \leftarrow$ **find step length**$(0,\mathbf{X})$ (**Procedure** 2)
    $\mathbf{w} \leftarrow$ **importance weights**$(\alpha,\pi,\mathbf{X})$ (**Procedure** 1)
  while $\rho < 1$ **do**
    $q_\theta \leftarrow$ **fit binary model**$(w,\mathbf{X})$ (**Section** 5)
    $\widehat{\mathbf{X}} \leftarrow$ **resample**$(w,\mathbf{X})$ (**Procedure** 3)
    $\mathbf{X} \leftarrow$ **move**$(\kappa_{\pi,q_\theta},\widehat{\mathbf{X}})$ (**Procedure** 4)
    $\alpha \leftarrow$ **find step length**$(\rho,\mathbf{X})$ (**Procedure** 2)
    $\mathbf{w} \leftarrow$ **importance weights**$(\alpha,\pi,\mathbf{X})$ (**Procedure** 1)
    $\rho \leftarrow \rho + \alpha$
  end while
  return $\sum_{k=1}^n w_k \mathbf{x}_k \approx \mathbb{E}_\pi(\gamma)$

can exploit the fact that, in a systematically resampled particle system, multiple copies of the same particles are neighbours.

5 Multivariate binary models

In the section, we address the choice of a multivariate binary parametric family $\{q_\theta \mid \theta \in \Theta\}$ needed to construct the Metropolis-Hastings kernel used in Procedure 4.

5.1 Desired properties

We first frame the properties making a parametric family suitable for our Sequential Monte Carlo algorithm.

(a) For reasons of parsimony, we want to construct a family of distributions with at most $\dim(\theta) \le d(d+1)/2$ parameters. More complex families are usually computationally too expensive to handle.

(b) Given a sample $\mathbf{X} = (\mathbf{x}_1,\dots,\mathbf{x}_n)$ from the target distribution π, we want to estimate θ^* such that the binary model q_{θ^*} is close to π. For instance, θ^* might be a maximum likelihood or method of moments estimator.

(c) We want to generate independent samples from q_θ. If we can compute the conditional or marginal distributions, we can write q_θ as

$$q_\theta(\gamma) = q_\theta(\gamma_1)\prod_{i=2}^d q_\theta(\gamma_i|\gamma_{1:i-1}) \tag{11}$$
$$= q_\theta(\gamma_1)\prod_{i=2}^d q_\theta(\gamma_{1:i})/q_\theta(\gamma_{1:i-1}).$$

Using the chain rule decomposition (11), we can sample a random vector $\gamma \sim q_\theta$ component-wise, conditioning on the entries we already generated.

(d) We need to rapidly evaluate $q_\theta(\gamma)$ for any $\gamma \in \mathbb{B}^d$ in order to compute the Metropolis-Hastings ratio (5).

(e) Analogously to the multivariate normal, we want our calibrated binary model q_{θ^*} to produce samples with the mean and covariance of π. If q_θ is not flexible enough to capture the dependence structure of π, the Metropolis-Hastings kernel in Procedure 4 cannot provide satisfactory acceptance rates for complex target distributions π.

In the following we construct a suitable parametric family and explain how to deploy it in Algorithm 2.

Most of the literature on binary data stems from response models, multi-way contingency tables and multivariate interaction theory (Cox, 1972). For completeness, we append a brief list of other binary models mentioned in the literature which fail, for various reasons, to work in Sequential Monte Carlo applications. Providing parametric families which meet the above requirements in high dimensions is a difficult task and understanding the shortcomings of alternative approaches an important part of the discussion.

5.2 The logistic regression model

In the previous paragraph, we already mentioned that a factorization (11) of the mass function $q_\theta(\gamma)$ into conditional distributions permits to sample from the parametric family. Unfortunately, for a complex d-dimensional binary model, we usually cannot calculate closed-form expressions for the conditional or marginal mass functions.

Construction of the model We get around computing the marginal distributions of $q_\theta(\gamma)$ if we directly fit univariate models $q_{\mathbf{b}_i}(\gamma_i \mid \gamma_{1:i-1})$ to the conditionals $\pi(\gamma_i \mid \gamma_{1:i-1})$ of the target function. Precisely, we adjust the logistic regressions

$$\operatorname{logit}(\mathbb{P}_\pi(\gamma_i = 1)) \overset{def}{=} b_{i,i} + \textstyle\sum_{j=1}^{i-1} b_{i,j}\gamma_j, \quad i = 1,\dots,d$$

where $\operatorname{logit}(p) = \log p - \log(1-p)$. In the context of our Sequential Monte Carlo application, we take the particle system $\mathbf{X}$ and regress $\mathbf{y}^{[i]} = \mathbf{X}_i$ on the columns $\mathbf{Z}^{[i]} = (\mathbf{X}_{1:i-1}, \mathbf{1})$, where the column $\mathbf{Z}_i^{[i]}$ yields the intercept to complete the logistic model.

For a d-dimensional lower triangular matrix $\mathbf{B}$, we define the logistic regression model as

$$q_{\mathbf{B}}(\gamma) \overset{def}{=} \prod_{i=1}^d \mathscr{B}_{p(b_{i,i}+\mathbf{b}_{i,1:i-1}\gamma_{1:i-1}^\intercal)}(\gamma_i) \tag{12}$$

where $p(y) = \operatorname{logit}^{-1}(y) = (1+\exp(-y))^{-1}$. As in the preceding sections, $\mathscr{B}_p(\gamma) = p^\gamma(1-p)^{1-\gamma}$ is the Bernoulli distribution with parameter $p \in [0,1]$.

There are $d!$ possible logistic regressions models and we arbitrarily pick one while there should be a parametrization which is optimal in a sense of nearness to the data $\mathbf{Z}$. We observed, however, that permuting the components had, in practice, no impact on the quality of the approximation.

Keep in mind that the number of observations in the logistic regressions is the size n of the particle system and typically very large. For instance, we run our numerical examples in Section 6 using $n = 2 \times 10^6$ particles. Therefore, the fit of the logistic regressions is usually very good.

Sparse version of the model The major drawback of all multiplicative models is the fact that we have no closed-form likelihood-maximizers such that the parameter estimation requires costly iterative fitting procedures. Therefore, even before discussing the fitting procedure, we construct a sparse version of the logistic regression model which we can estimate faster than the saturated model.

Instead of fitting the saturated model $q(\gamma_i \mid \gamma_{1:i-1})$, we preferably work with a more parsimonious regression model like $q(\gamma_i \mid \gamma_{L_i})$ for some index set $L_i \subseteq \{1,\dots,i-1\}$, where the number of predictors $\#L_i$ is typically smaller than $i-1$. We solve this nested variable selection problem using some simple, fast to compute criteria.

Given a weighted particle system $w \in [0,1]^n$, $\mathbf{X} \in \mathbb{B}^{n\times d}$, we denote for $i,j \in \{1,\dots,d\}$ the weighted sample mean by

$$\bar{x}_i = \textstyle\sum_{k=1}^n w_k x_{k,i}, \quad \bar{x}_{i,j} = \sum_{k=1}^n w_k x_{k,i} x_{k,j}, \tag{13}$$

and the weighted sample correlation by

$$r_{i,j} = \frac{\bar{x}_{i,j} - \bar{x}_i \bar{x}_j}{\sqrt{\bar{x}_i(1-\bar{x}_i)\bar{x}_j(1-\bar{x}_j)}}. \tag{14}$$

For $\varepsilon = 0.02$, we define the index set

$$I \stackrel{def}{=} \{i = 1,\dots,d \mid \bar{x}_i \notin (\varepsilon, 1-\varepsilon)\}.$$

which identifies the components which have, according to particle system, a marginal probability close to either boundary of the unit interval.

For the components $i \in I$, we do not consider fitting a logistic regression, but set $L_i = \emptyset$ and draw them independently. Precisely, we set $b_{i,i} = \mathrm{logit}(\bar{x}_i)$ and $\mathbf{b}_{i,-i} = \mathbf{0}$ which corresponds to logistic model without predictors. Firstly, interactions do not really matter if the marginal probability is excessively small or large. Secondly, these components are prone to cause complete separation in the data or might even be constant.

For the conditional distribution of the remaining components $I^c = \{1,\dots,d\} \setminus I$, we construct parsimonious logistic regressions. For $\delta = 0.075$, we define the predictor sets

$$L_i \stackrel{def}{=} \{j = 1,\dots,i-1 \mid \delta < |r_{i,j}|\}, \quad i \in I^c,$$

which identifies the components with index smaller than i and significant mutual association. Running our examples in Section 6 with $\delta = 0$ show that a saturated logistic regression kernel achieves about the same acceptance rates as a sparse one, while setting $\delta = 0.075$ dramatically reduces the computational time we need to calibrate the model.

Fitting the model We maximise the log-likelihood function $\ell(b) = \ell(\mathbf{b} \mid \mathbf{y}, \mathbf{Z})$ of a weighted logistic regression model by solving the first order condition $\partial\ell/\partial\beta = \mathbf{0}$. We find a numerical solution via Newton-Raphson iterations

$$-\frac{\partial^2 \ell(\mathbf{b}^{[r]})}{\partial \mathbf{b}\mathbf{b}^\intercal}(\mathbf{b}^{[r+1]} - \mathbf{b}^{[r]}) = \frac{\partial \ell(\mathbf{b}^{[r]})}{\partial \mathbf{b}}, \quad r > 0, \tag{15}$$

starting at some $\mathbf{b}^{[0]}$; see Procedure 5 for the exact terms. Other updating formulas like Iteratively Reweighted Least Squares or quasi-Newton iterations should work as well.

Procedure 5 Fitting the weighted logistic regressions

Input: $\mathbf{w} = (w_1,\dots,w_n)$, $\mathbf{X} = (\mathbf{x}_1,\dots,\mathbf{x}_n)^\intercal$, $\mathbf{B} \in \mathbb{R}^{d\times d}$
  for $i \in I^c$ **do**
    $\mathbf{Z} \leftarrow (\mathbf{X}_{L_i}, \mathbf{1})$, $\mathbf{y} \leftarrow \mathbf{X}_i$, $\mathbf{b}^{[0]} \leftarrow \mathbf{B}_{i,L_i\cup\{i\}}$
    repeat
      $p_k \leftarrow \mathrm{logit}^{-1}(\mathbf{Z}_k \mathbf{b}^{[r-1]})$ **for all** $k = 1,\dots,n$
      $q_k \leftarrow p_k(1-p_k)$ **for all** $k = 1,\dots,n$

      $\mathbf{b}^{[r]} \leftarrow (\mathbf{Z}^\intercal \mathrm{diag}[\mathbf{w}]\,\mathrm{diag}[\mathbf{q}]\,\mathbf{Z} + \varepsilon \mathbf{I}_n)^{-1} \times$
        $(\mathbf{Z}^\intercal \mathrm{diag}[\mathbf{w}])\left(\mathrm{diag}[\mathbf{q}]\,\mathbf{Z}\mathbf{b}^{[r-1]} + (\mathbf{y} - \mathbf{p})\right)$
    until $|b_j^{[r]} - b_j^{[r-1]}| < 10^{-3}$ for all j
    $\mathbf{B}_{i,L_i\cup\{i\}} \leftarrow \mathbf{b}$
  end for
  return $\mathbf{B}$

Sometimes, the Newton-Raphson iterations do not converge because the likelihood function is monotone and thus has no finite maximizer. This problem is caused by data with complete or quasi-complete separation in the sample points (Albert and Anderson, 1984). There are several ways to handle this issue.

(a) We just halt the algorithm after a fixed number of iterations and ignore the lack of convergence. Such proceeding, however, might cause uncontrolled numerical problems.

(b) Firth (1993) proposes to use a Jeffrey's prior on $\mathbf{b}$. The penalized log-likelihood does have a finite maximizer but requires computing the derivatives of the Fisher information matrix.

(c) We just add a simple quadratic penalty term $\varepsilon\beta^\intercal\beta$ to the log-likelihood to ensure the target-function is convex and does not cause numerical problems.

(d) As we notice that some terms of $\mathbf{b}_i$ are growing beyond a certain threshold, we move the component i from the set of components with associated logistic regression model I^c to the set of independent components I.

In practice, we combine the approaches (c) and (d). In Procedure 5, we did not elaborate how to handle non-convergence, but added a penalty term to the log-likelihood,

which causes the extra $\varepsilon \mathbf{I}_n$ in the Newton-Raphson update. Since we solve the update equation via Cholesky factorizations, adding a small term on the diagonal ensures that the matrix is indeed numerically decomposable.

Starting points The Newton-Raphson procedure is known to rapidly converge for starting values $b_i^{[0]}$ not too far from the solution $b_i^{[*]}$. In absence of prior information about $b_i^{[*]}$, we would naturally start with a vector of zeros and maybe setting $b_{i,i}^{[0]} = \text{logit}(\overline{x_i})$.

In the context of our Sequential Monte Carlo algorithm we can do better than that. Recall that, we constructed a smooth sequence $(\pi_t)_{t=0}^{\tau}$ of distributions which corresponds to a sequence of proposal distributions $(q_t)_{t=0}^{\tau} = (q_{\theta_t})_{t=0}^{\tau}$, which is associated to a sequence $(\theta_t)_{t=0}^{\tau}$ of parameters.

It significantly speeds up the Newton-Raphson procedure if we choose $\theta_t = \mathbf{B}_t$ as starting point for the estimation of $\mathbf{B}_{t+1}$. Indeed, towards the end of the Sequential Monte Carlo algorithm, we fit a single next logistic regression $\text{logit}(X_i) \sim X_{L_i}$ in less than four iterations on average when starting at $\mathbf{B}_{t-1}$, compared to about 13 iterations on average when starting at zero.

Sampling and evaluating In the move step of Sequential Monte Carlo we discussed in Section 4.5, we need to sample a proposal state $\mathbf{y}$ from q_θ and evaluate the likelihood $q_\theta(y)$ to compute the Metropolis-Hastings ratio 5. For the logistic regression model $q_{\mathbf{B}}$, we can do both in one go, see Procedure 6.

Procedure 6 Sampling from the model

Input: $\mathbf{B}$
  $\mathbf{y} \leftarrow (0,\ldots,0),\ p \leftarrow 1$
  for $i = 1\ldots,d$ **do**
    $q \leftarrow \text{logit}^{-1}(b_{i,i} + \sum_{j\in L_i} b_{i,j} y_j)$
    sample $\gamma_i \sim \mathscr{B}_q$
    $p \leftarrow \begin{cases} p \times q & \textbf{if}\ \ y_i = 1 \\ p \times (1-q) & \textbf{if}\ \ y_i = 0 \end{cases}$
  end for
  return $\mathbf{y},\ p$

5.3 Why not use a simpler model?

We briefly justify why we should not use a simpler parametric family for our Sequential Monte Carlo application. Indisputably, the easiest parametric family on $\mathbb{B}^d$ that we can think of is a product model

$$q_{\mathbf{p}}(\gamma) \stackrel{def}{=} \prod_{i=1}^{d} \mathscr{B}_{p_i}(\gamma_i)$$

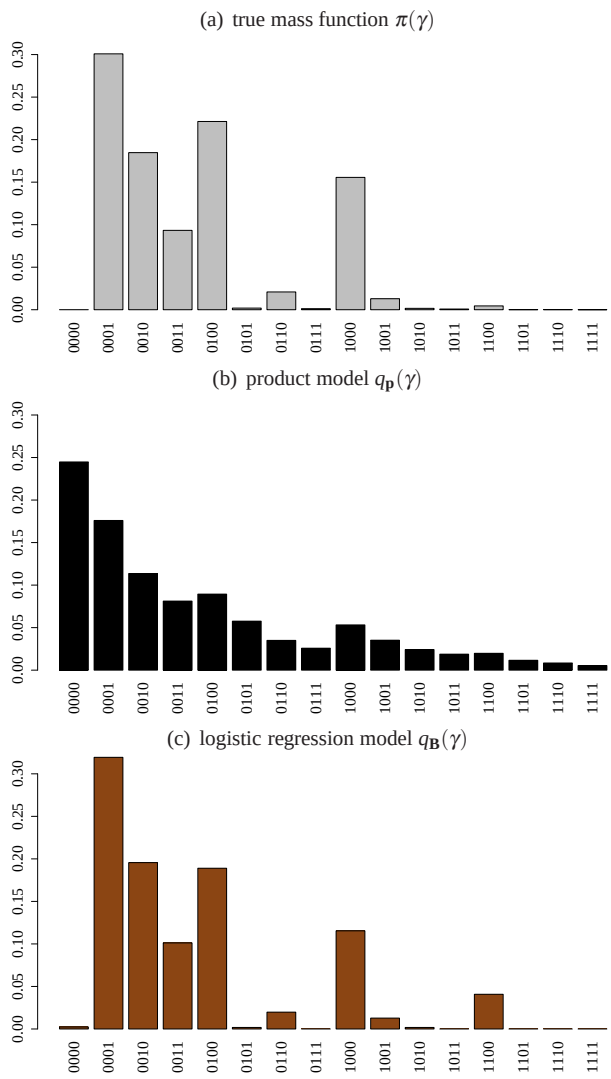
where $\mathscr{B}_{p_i(x)}(\gamma) = p_i(x)^{\gamma}(1-p_i(x))^{1-\gamma}$ denotes a Bernoulli distribution with parameter $p_i(x) \in [0,1]$.

Let us check the requirement list: the product model is parsimonious with $\dim(\theta) = d$; the maximum likelihood estimator θ^* is the sample mean $\overline{\mathbf{x}} = n^{-1}\sum_{k=1}^{n} x_k$; the decomposition (11) holds trivially, which allows us to sample from $q_{\mathbf{p}}$ and evaluate $q_{\mathbf{p}}(\gamma)$ in $\mathrm{O}(d)$.

Obviously, however, $q_{\mathbf{p}}$ does not reproduce any dependencies we might observe in $\mathbf{X}$. Could we just forget about this last point and use the product model for its simplicity?

Toy example We visualize the strinking impracticalness of the product model by means of a toy example in a low dimension $d = 3$. We take a simple linear relation $\mathbf{Y} = \mathbf{V}_1 + \mathbf{V}_2$ and construct a variable selection problem with high dependencies.

Figure 1: Toy example showing how well the product model $q_{\mathbf{p}}$ and the logistic regression model $q_{\mathbf{B}}$ replicate the mass function of a difficult posterior distribution π.



For $n = 100$ and $\mu = 10$, we draw normal variates

$$\mathbf{v}_1 \sim \mathcal{N}(-\mu, \mathbf{I}_n), \quad \mathbf{v}_2 \sim \mathcal{N}(\mu, \mathbf{I}_n), \quad \mathbf{y} = \mathbf{v}_1 + \mathbf{v}_2$$

and then generate observations

$$z_1, z_2 \sim \mathcal{N}\left(w_1, (\mu^2/4)\mathbf{I}_n\right), \quad z_3, z_4 \sim \mathcal{N}\left(w_2, (\mu^2/4)\mathbf{I}_n\right).$$

The posterior distribution $\pi(\gamma) = \pi(\gamma \,|\, \mathbf{y}, \mathbf{Z})$, using the prior distributions as described in Section 2, typically exhibits strong dependencies between its components due to the correlation in the data.

Now we generate pseudo-random data $\mathbf{X}$ from π and fit both a product model $q_{\mathbf{p}}$ and a logistic regression model $q_{\mathbf{B}}$. Looking at the corresponding mass function in Figure 1, we notice how badly the product model mimics the true posterior. This observation carries over to larger sampling spaces.

Acceptance rates A good way to analyse the importance of reproducing the dependencies of π is in terms of acceptance rates and particle diversities. As we already remark in Section 4.5, the particle diversity naturally diminishes as our particle system approaches a strongly multi-modal target distribution π. However, we want our algorithm to keep up the particle diversity a long as possible to ensure the particle system is well spread out over the entire state space.

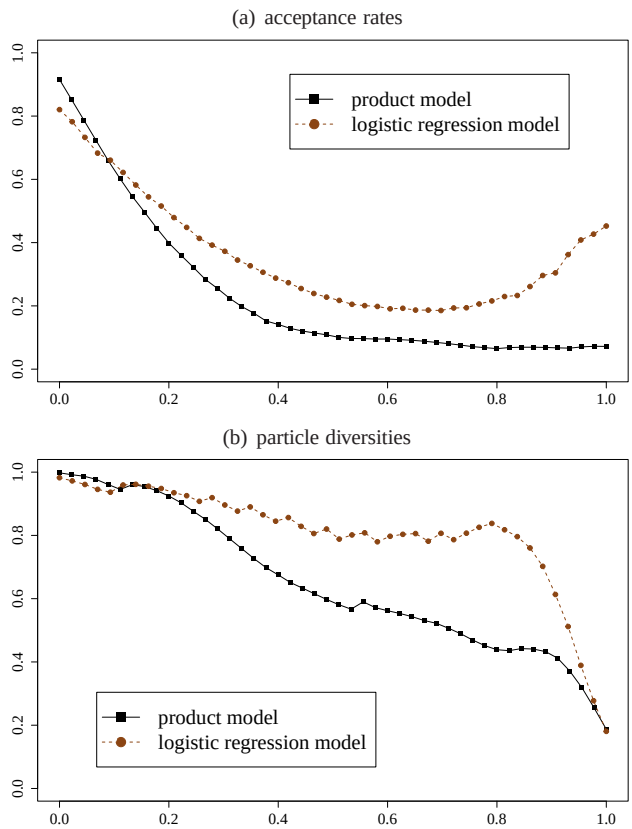
In Figure 2, we show a comparison (based on the Boston Housing data set explained in Section 6.1) between two Sequential Monte Carlo algorithms, using a product model $q_{\mathbf{p}}$ and a logistic regression model $q_{\mathbf{B}}$ as proposal distribution of the Metropolis-Hastings kernel (3.4).

Clearly, in Figure 2(a), the acceptance rates achieved by the product kernel rapidly decrease and dwell around 5% for the second half of the run. In contrast, the logistic regression kernel always provides acceptance rates greater than 20%. As a consequence, in Figure 2(b), the particle diversity sustained by the product kernel decreases at an early stage, while the logistic regression kernel holds it up until the very last steps.

At first sight, it might seem odd that the acceptance rates of the logistic regression kernel increase during the final steps of the algorithm. If we jump ahead, however, and take a look at the results of the Boston Housing problem, see Figure 3(a), we notice that quite a few marginal probabilities of the posterior π turn out to be zero, which makes it easier to reproduce the distributions towards the end of the Resample-Move algorithm.

However, if we already decide at an early stage that for some component $\mathbb{P}(\gamma_i = 1) = 0$, we fail to ever consider states $\gamma \in \mathbb{B}^d$ with $\gamma_i = 1$ for the rest of the algorithm. Therefore, the advantage of the logistic regression kernel over the simple product kernel is that we do not completely drop any components from the variable selection problem until the final steps.

Figure 2: We compare the use of a product model $q_{\mathbf{p}}$ to a logistic regression model $q_{\mathbf{B}}$ as proposal distribution of the Metropolis-Hastings kernel (3.4). We monitor a typical run (ρ on the x-axis) of our Sequential Monte Carlo algorithm (for the Boston Housing data set described in Section 6.1) and plot the acceptance rates and particle diversities (on the y-axis).



5.4 Review of alternative binary models

In the following, we review some alternative approaches to modeling multivariate binary data. Unfortunately, we cannot incorporate any of these models in our Sequential Monte Carlo algorithm. Still, it is instructive to understand why alternative strategies fail to provide suitable proposal distributions in the sense of Section 5.1. For a more detailed review of parametric families suitable for adaptive Monte Carlo algorithms on binary spaces, see Schäfer (2010).

Additive models For suitable coefficients $a \in \mathbb{R}^{2^d}$, we can write any mass function on $\mathbb{B}^d$ as

$$\pi(\gamma) = \textstyle\sum_{S \subseteq \{1,\ldots,d\}} a_S \prod_{i \in S} \gamma_i.$$

It is tempting to construct a $d(d+1)/2$ parameter model

$$q_{\mu,\mathbf{A}}(\gamma) \stackrel{def}{=} \mu + \gamma^{\mathsf{T}} \mathbf{A} \gamma$$

by removing interaction terms of order higher than two. As Bahadur (1961) points out, the main problem of any additive

approach is the fact that a truncated model might not be non-negative and thus not define a probability distribution.

Although the linear structure allows to derive explicit and recursive formulae for the marginal and conditional distributions, we hardly ever find a useful application for the additive model. As other authors (Park et al., 1996; Emrich and Piedmonte, 1991) remark, additive representations like the much-cited Bahadur (1961) expansion are quite instructive but, unfortunately, impractical.

Log-linear models For suitable coefficients $a \in \mathbb{R}^{2^d}$, we can write any mass function on $\mathbb{B}^d$ as

$$\pi(\gamma) = \exp\left(\textstyle\sum_{S \subseteq \{1,\dots,d\}} a_S \prod_{i \in S} \gamma_i.\right)$$

Removing higher order interaction terms, we can construct a $d(d+1)/2$ parameter model

$$q_{\mu,\mathbf{A}}(\gamma) \stackrel{def}{=} \mu \exp(\gamma^\intercal \mathbf{A} \gamma), \tag{16}$$

where $\mathbf{A}$ is a symmetric matrix. Log-linear models define a well studied class of distributions, but there is no simple recursive structure for their marginal distributions. Therefore, we cannot compute the factorization (11) we need to sample from $q_{\mathbf{A}}$.

Cox and Wermuth (1994) propose an approximation to the marginal distributions by expressions of the form (16), omitting higher order terms in a Taylor expansion. If we write the parameter $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' & \mathbf{b}^\intercal \\ \mathbf{b} & c \end{pmatrix},$$

the parameter of the marginal distribution $q_{\mathbf{A}_{1:d-1}}(\gamma_{1:d-1})$ is approximately given by

$$\mathbf{A}_{1:d-1} \approx \mathbf{A}' + (1+\tanh(c/2))\,\mathrm{diag}\,[\mathbf{b}] + \frac{1}{2}\mathrm{sech}^2(c/2)\mathbf{b}\mathbf{b}^\intercal,$$

which is a symmetric matrix of size $d-1$. The normalization constant is $\mu_{1:d-1} = \mu(1+\exp(c))$. We can recursively compute approximations to all marginal distributions $q_{\mathbf{A}_{1:d-1}}, \dots, q_{\mathbf{A}_{1:1}}$ and derive logistic forms

$$\mathrm{logit}(\mathbb{P}(\gamma_i = 1 \mid \gamma_{1:i-1})) = \log \frac{q_{A_{1:i}}(\gamma_i = 1, \gamma_{1:i-1})}{q_{A_{1:i}}(\gamma_i = 0, \gamma_{1:i-1})},$$

which takes us back to (12). However, there is no reason to fit a log-linear model and compute approximate logistic models if we can directly fit a logistic regression model in the same time.

Latent variable models Let p_θ be a parametric family on $\mathscr{X}$ and $s\colon \mathscr{X} \to \mathbb{B}^d$ a mapping into the binary state space. We can sample from a latent variable model

$$q_\theta(\gamma) = \textstyle\int_{s^{-1}(\gamma)} p_\theta(v)\, d\mathbf{v}$$

by setting $\mathbf{y} = s(\mathbf{v})$ for a draw $\mathbf{v} \sim p_\theta$ from the latent parametric family.

Note that evaluating the probability mass function $q_\theta(\mathbf{y})$ is usually a difficult task. Hence, we cannot use this class of models in a Sequential Monte Carlo context. These models can be useful, however, in other adaptive Monte Carlo algorithms that do not require evaluation of $q_\theta(\mathbf{y})$, for instance the Cross-Entropy method (Rubinstein, 1997).

Non-normal parametric families with $d(d-1)/2$ dependence parameters seem to either have a very limited dependence structure or unfavourable properties (Joe, 1996). Therefore, the multivariate normal

$$p_{(\mu,\Sigma)}(\mathbf{v}) = (2\pi)^{-d/2}|\Sigma|^{-1/2} e^{-1/2(v-\mu)^\intercal \Sigma^{-1}(v-\mu)},$$
$$s(\mathbf{v}) = (\mathbb{1}_{(\infty,0]}(v_1), \dots, \mathbb{1}_{(\infty,0]}(v_d)),$$

appears to be the natural and almost the only option for p_θ. This kind of model has been discussed repeatedly in the literature (Emrich and Piedmonte, 1991; Leisch et al., 1998; Cox and Wermuth, 2002).

The first and second order marginal probabilities of the model $q_{(\mu,\Sigma)}$ are given by $\Phi_1(\mu_i)$ and $\Phi_2(\mu_i, \mu_j; \sigma_{i,j})$, respectively, where $\Phi_1(v_i)$ and $\Phi_2(v_i, v_j; \sigma_{i,j})$ denote the cumulative distribution functions of the univariate and bivariate normal distributions with zero mean, unit variance and correlation $\sigma_{i,j} \in [-1, 1]$.

We can fit the model $q_{(\mu,\Sigma)}$ to a particle system $(\mathbf{w}, \mathbf{X})$ by matching the moment, that is adjusting μ and Σ such that

$$\Phi_1(\mu_i) = \bar{x}_i, \quad \Phi_1(\mu_i, \mu_j; \sigma_{i,j}) = r_{i,j}$$

with $\bar{x}_i$ and $r_{i,j}$ as defined in (13) and (14). However, the locally constructed correlation matrix Σ might not be positive definite. Still, we can obtain a feasible parameter replacing Σ by $\Sigma^* = (\Sigma + |\lambda|\mathbf{I})/(1+|\lambda|)$, where λ is the smallest eigenvalue of the locally adjusted matrix Σ.

Archimedean copula models Genest and Neslehova (2007) discuss the potentials and pitfalls of applying copula theory, which is well developed for bivariate, continuous random variables, to multivariate discrete distribution. There have been earlier attempts to sample binary vectors via copulae: Lee (1993) describes how to construct an Archimedean copula, more precisely the Frank family (Nelsen, 2006, p.119), for sampling multivariate binary data. Unfortunately, this approach is limited to very low dimensions.

Multivariate reduction models Several approaches to generating multivariate binary data are based on a representation of the components γ_i as functions of sums of independent variables (Park et al., 1996; Lunn and Davies, 1998; Oman and Zucker, 2001). These techniques are limited to certain patterns of non-negative correlation, and do, therefore, not yield suitable proposal distributions in a Sequential Monte Carlo application. We mention them for the sake of completeness.

6 Numerical examples

In this section we compare our Sequential Monte Carlo algorithm to standard Markov chain methods. We created model choice problems with high dependencies between the covariates which yield particularly challenging, multi-modal posterior mass functions. Our examples are build from freely available datasets by adding logarithms, polynomials and interaction terms.

6.1 Construction of the data sets

Boston Housing The first example is based on the Boston Housing data set, originally treated by Harrison and Rubinfeld (1978), which is freely available at the StatLib data archive. The data set provides covariates ranging from the nitrogen oxide concentration to the per capita crime rate to explain the median prices of owner-occupied homes. The data has yet been treated by several authors, mainly because it provides a rich mixture of continuous and discrete variables, resulting in an interesting variable selection problem.

Specifically, we aim at explaining the logarithm of the corrected median values of owner-occupied housing. We enhance the 13 columns of the original data set by adding first order interactions between all covariates. Further, we add a constant column and a squared version of each covariate (except for CHAS, since it is binary).

This gives us a model choice problem with 104 possible predictors and 506 observations. We use a hierarchical Bayesian approach, with priors as explained in the above Section 2, to construct a posterior distribution π. By construction, there are strong dependencies between the possible predictors which leads to a rather complex, multi-modal posterior distribution.

Concrete Compressive Strength The second example is constructed from a less known data set, originally treated by Yeh (1998), which is freely available at the UCI Machine Learning Repository. The data provides information about composing concrete to explain its compressive strength. The compressive strength appears to be a highly non-linear function of age and ingredients.

In order to explain the compressive strength, we take the 8 covariates of the original data set and add the logarithms of some covariates (cement, water, coarse aggregate, fine aggregate, age). Further, we add interactions between all 13 covariates of the augmented data set and a constant column.

This gives us a model choice problem with 79 possible predictors and 1030 observations. We use a hierarchical Bayesian approach, with priors as explained in the above Section 2, to construct a posterior distribution π.

6.2 How to compare to Markov chain Monte Carlo

We do not think it is reasonable to compare two completely different algorithms in terms of pure computational time. We cannot guarantee that our implementations are optimal nor that the time measurements can exactly be reproduced in other computing environments.

We suppose that the number of evaluations of the target function π is more of a fair stopping criterion, since it shows how well the algorithms exploit the information obtained from π. Precisely, we parameterise the Sequential Monte Carlo algorithm to not exceed a fixed number ν of evaluations and stop the Markov chains when ν evaluations have been performed.

Assets and drawbacks The Sequential Monte Carlo and the Markov chain Monte Carlo algorithms both have extensions and numerical speed-ups which make it hard to settle on a fair comparison.

Advocates of Markov chain methods might criticise that the number of target evaluations is a criterion biased towards the Sequential Monte Carlo approach, for there are updating schemes which allow for faster computation of the Cholesky decomposition (3) given the decomposition of a neighbouring model, see Dongarra et al. (1979, chaps. 8,10). Thus, Markov chains which propose to change one component in each step can evaluate π with less effort and perform more evaluations of π in the same time.

On the other hand, however, the Sequential Monte Carlo algorithm can be parallelised in the sense that we can, on suitable hardware, run many evaluations of π in parallel during the move step, see Procedure 4. No analogue speed-up can be performed in the context of Markov chains. We did not yet exploit this advantage but are confident that we shall see this feature in a follow-up of this paper. Further, Sequential Monte Carlo methods are more suitable than Markov chain Monte Carlo to approximate the evidence, that is the normalization constant of the posterior distribution. We can exploit this property to compare, for instance, regression models with different monotonic link functions.

Parameters We run our Sequential Monte Carlo (SMC) algorithm with $n = 2 \times 10^4$ particles and a target effective sample size $\eta = 0.9$, as explained in Section 4. For these parameters, the Sequential Monte Carlo algorithm needs less than $\nu = 2 \times 10^6$ evaluations of π on both examples problems.

We compare our algorithm to both the Adaptive Markov chain Monte Carlo (Nott and Kohn, 2005, AMCMC) and the standard metropolised Gibbs (Liu, 1996, MCMC) described in Section 3. As stated earlier, we could not observe any positive effect from block updating and do therefore not consider it in our examples.

For the AMCMC, we use $\delta = 0.01$ and $\lambda = 0.01$, following the recommendations of Nott and Kohn (2005). We update the estimates ψ and $\mathbf{W}$ every 2×10^5 iterations of

chain. Before we start adapting, we generate 2×10^5 iterations with a metropolised Gibbs kernel (after a discarded burn-in of 2×10^4 iterations).

6.3 Implementation

The numerical work was completely done in Python 2.6 using SciPy packages. Scientific work in applied fields is often more accessible to the reader if the source code which generated numerical evidence is released along with the publication. The complete sources used in this work can be found at `http://code.google.com/p/smcdss`.

We also provide instructions on how to install and run our project. The program can process data sets in standard csv-format and generate R scripts for graphical visualisation of the results. The released version was tested to run on both Windows and Linux machines.

6.4 Discussion and conclusion

We run each algorithm 200 times and visualize the variation of the results in box-plots, see Figures 3 and 4. The white boxes on contain 80% of the results, while the black boxes contain the 20% outliers. The horizontal line in the white box indicates the median. We draw a coloured bar below the minimal value to improve the readability; otherwise components with a small variation are hard to see.

The Sequential Monte Carlo algorithm ist extremely robust. For 200 test runs and for both data sets, the algorithm did not produce a single outlier in any of the components.

This not true for either of the Markov chain algorithms. The size of white boxes indicate that adaptive Markov chain Monte Carlo works quite better than the standard Markov chain procedure. However, even the adaptive chain is rather vulnerable to outliers. The large black boxes indicate that, for some starting points of the chain, the estimates of some marginal probabilities might be completely wrong.

The outliers, that is the black boxes, in Figures 4(b) and 4(c) are strikingly similar. The adaptive and the standard Markov chains apparently both fall into the same trap, which in turn implies that adaption makes the method faster but not more robust against outliers. An adapted local method is still a local method and does not yield reliable estimates for difficult sampling problems.

In Tables 1 and 2, we gathered some key performance indicators, each averaged over the 200 runs of the respective algorithms. Note that the time needed to perform 2×10^6 evaluations of π is a little less than the running time of the standard Markov chain. Thus, even in terms of computational time, the adaptive Markov chain can hardly compete with our Sequential Monte Carlo method, even if evaluations of π were at no cost.

Acknowledgements N. Chopin is supported by the ANR grant ANR-008-BLAN-0218 “BigMC” of the French Ministry of research.

We acknowledge the StatLib data archive and the UCI Machine Learning Repository for providing the data sets used in this work.

References

Albert, A. and Anderson, J. A. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, (72):1–10.

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

Bahadur, R. (1961). A representation of the joint distribution of responses to n dichotomous items. In Solomon, H., editor, *Studies in Item Analysis and Prediction*, pages pp. 158–68. Stanford University Press.

Cappé, O., Douc, R., Guillin, A., Marin, J., and Robert, C. (2008). Adaptive importance sampling in general mixture classes. *Statistics and Computing*, 18(4):447–459.

Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Improved particle filter for nonlinear problems. *IEE Proc. Radar, Sonar Navigation*, 146(1):2–7.

Cox, D. (1972). The analysis of multivariate binary data. *Applied Statistics*, pages 113–120.

Cox, D. and Wermuth, N. (1994). A note on the quadratic exponential binary distribution. *Biometrika*, 81(2):403–408.

Cox, D. and Wermuth, N. (2002). On some models for multivariate binary variables parallel in complexity with the multivariate Gaussian distribution. *Biometrika*, 89(2):462.

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.

Dongarra, J., Moler, C., Bunch, J., and Stewart, G. (1979). *LINPACK: users' guide*. Society for Industrial and Applied Mathematics.

Emrich, L. and Piedmonte, M. (1991). A method for generating high-dimensional multivariate binary variates. *The American Statistician*, 45(4):302–304.

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

Gelman, A. and Meng, X. (1998). Simulating normalizing constants: From importance sampling to bridge sampling to path sampling. *Statistical Science*, 13(2):163–185.

Genest, C. and Neslehova, J. (2007). A primer on copulas for count data. *Astin Bulletin*, 37(2):475.

Figure 3: Boston Housing data set. We ran the Sequential Monte Carlo, Adaptive Markov chain Monte Carlo and standard Markov chain Monte Carlo algorithms each 200 times. The white boxes contain 80% of the results with the line indicating the median. We added the coloured bars to make the plot easier to read.

(a) SMC $\sim 1.6 \times 10^6$ evaluations of π (b) AMCMC 2.0×10^6 evaluations of π (c) MCMC 2.0×10^6 evaluations of π

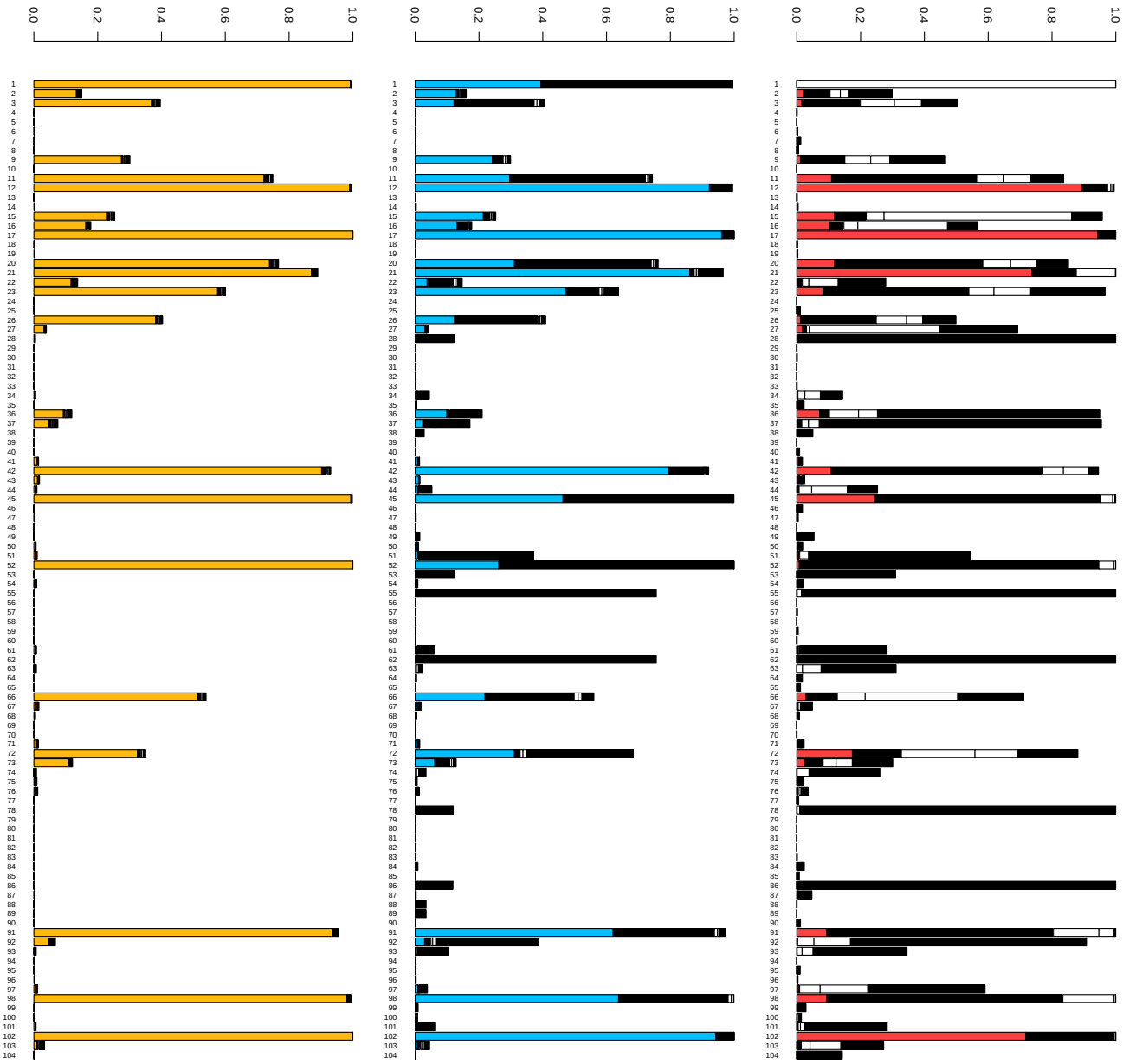


Table 1: Boston Housing data set. The table shows some averaged key indicators complementary to Figure 3

	Sequential MC	Adaptive MCMC	Standard MCMC
computational time	0 : 22 : 12 h	2 : 07 : 20 h	0 : 14 : 46 h
evaluations of π	1.91×10^6	2.00×10^6	2.00×10^6
average acceptance rate	36.4%	27.2%	0.02%
length t of the chain $\mathbf{x}_t$		5.39×10^7	2.00×10^6
moves $\mathbf{x}_t \neq \mathbf{x}_{t-1}$ of the chain		5.46×10^5	0.33×10^5

Figure 4: Concrete Compressive Strength data set. We ran the Sequential Monte Carlo, Adaptive Markov chain Monte Carlo and standard Markov chain Monte Carlo algorithms each 200 times. The white boxes contain 80% of the results with the line indicating the median. We added the coloured bars to make the plot easier to read.

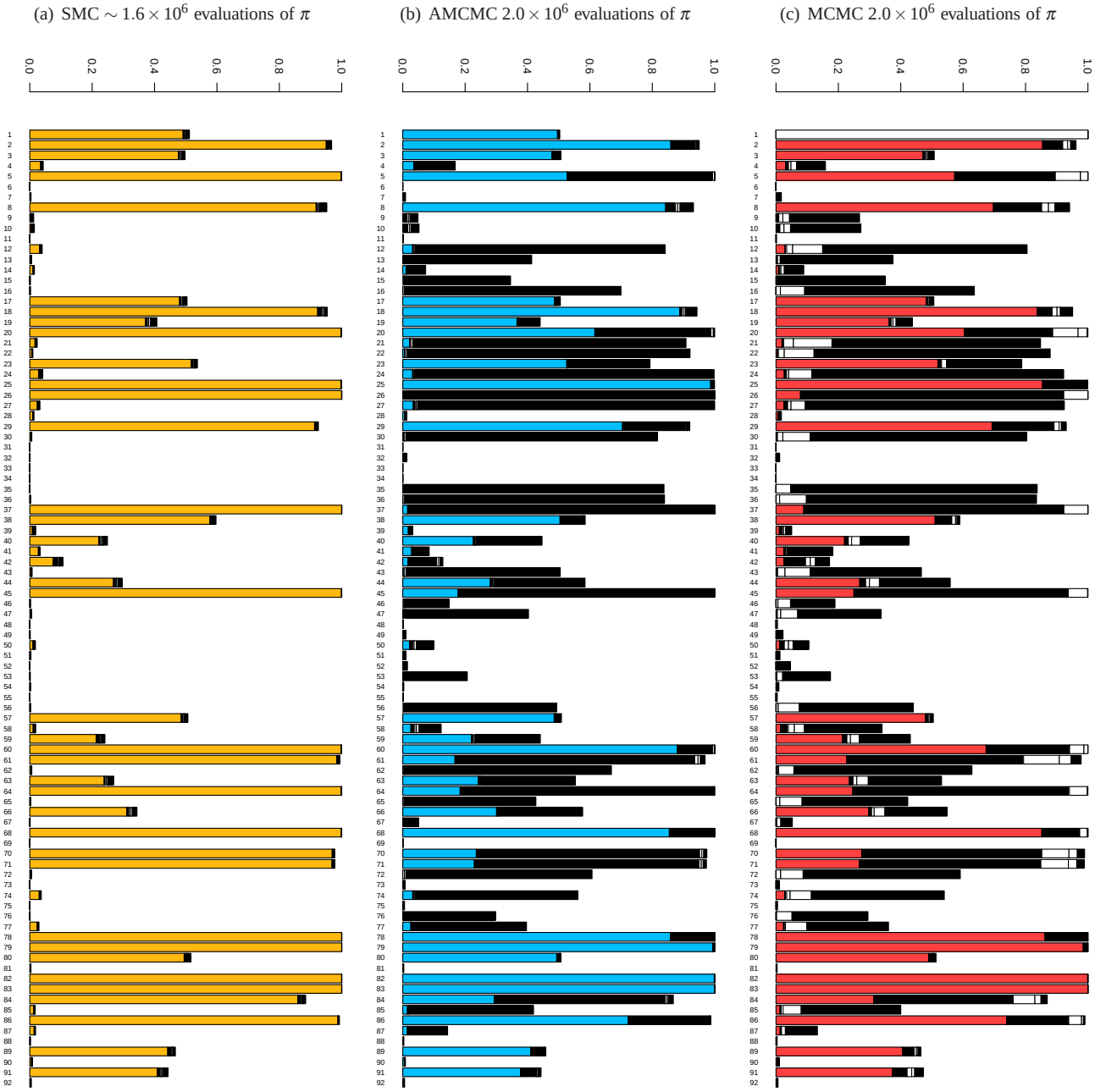


Table 2: Concrete Compressive Strength data set. The table shows some averaged key indicators complementary to Figure 4

	Sequential MC	Adaptive MCMC	Standard MCMC
computational time	20 : 54 min	53 : 45 min	19 : 18 min
evaluations of π	1.62×10^6	2.01×10^6	2.00×10^6
average acceptance rate	30.7%	67.1%	0.124%
length t of the chain $\mathbf{x}_t$		1.86×10^7	2.00×10^6
moves $\mathbf{x}_t \neq \mathbf{x}_{t-1}$ of the chain		1.35×10^5	0.25×10^5

George, E. I. and McCulloch, R. E. (1997). Approaches for Bayesian variable selection. *Statistica Sinica*, (7):339–373.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proc. F, Comm., Radar, Signal Proc.*, 140(2):107–113.

Harrison, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102.

Joe, H. (1996). Families of m-variate distributions with given margins and m (m-1)/2 bivariate dependence parameters. *Lecture Notes-Monograph Series*, 28:120–141.

Kitagawa, G. (1996). Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25.

Kong, A., Liu, J. S., and Wong, W. H. (1994). Sequential imputation and Bayesian missing data problems. *Journal of the American statistical association*, 89:278–288.

Lee, A. (1993). Generating Random Binary Deviates Having Fixed Marginal Distributions and Specified Degrees of Association. *The American Statistician*, 47(3).

Lee, A., Yau, C., Giles, M., Doucet, A., and Holmes, C. (2009). On the utility of graphics cards to perform massively parallel simulation of advanced Monte Carlo methods. *Arxiv preprint arXiv:0905.2441 v3*.

Leisch, F., Weingessel, A., and Hornik, K. (1998). On the generation of correlated artificial binary data. *preparation*.

Liu, J. (1996). Peskun's theorem and a modified discrete-state Gibbs sampler. *Biometrika*, 83(3):681–682.

Liu, J. and Chen, R. (1998). Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044.

Lunn, A. and Davies, S. (1998). A note on generating correlated binary variables. *Biometrika*, 85(2):487–490.

Neal, R. (2001). Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

Nelsen, R. (2006). *An introduction to copulas*. Springer Verlag.

Nott, D. and Kohn, R. (2005). Adaptive sampling for Bayesian variable selection. *Biometrika*, 92(4):747.

Oman, S. and Zucker, D. (2001). Modelling and generating correlated binary variables. *Biometrika*, 88(1):287.

Park, C., Park, T., and Shin, D. (1996). A Simple Method for Generating Correlated Binary Variates. *The American Statistician*, 50(4).

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

Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112.

Schäfer, C. (2010). Parametric families on large binary spaces. *Arxiv preprint arXiv:1008.0055*.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, (6):461–464.

Suchard, M., Holmes, C., and West, M. (2010). Some of the What?, Why?, How?, Who? and Where? of Graphics Processing Unit Computing for Bayesian Analysis. In Bernardo, J. M. et al., O. U. P., editor, *Bayesian Statistics 9*.

Yeh, I. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808.