Statistics and Computing (2024) 34:119
https://doi.org/10.1007/s11222-024-10404-w

ORIGINAL PAPER

Bayesian cross-validation by parallel Markov chain Monte Carlo

Alex Cooper[1] · Aki Vehtari[2] · Catherine Forbes[1] · Dan Simpson[3] · Lauren Kennedy[1,4]

Received: 23 October 2023 / Accepted: 8 February 2024 / Published online: 21 May 2024

Abstract

Brute force cross-validation (CV) is a method for predictive assessment and model selection that is general and applicable to a wide range of Bayesian models. Naive or ‘brute force’ CV approaches are often too computationally costly for interactive modeling workflows, especially when inference relies on Markov chain Monte Carlo (MCMC). We propose overcoming this limitation using massively parallel MCMC. Using accelerator hardware such as graphics processor units, our approach can be about as fast (in wall clock time) as a single full-data model fit. Parallel CV is flexible because it can easily exploit a wide range data partitioning schemes, such as those designed for non-exchangeable data. It can also accommodate a range of scoring rules. We propose MCMC diagnostics, including a summary of MCMC mixing based on the popular potential scale reduction factor ($\widehat{\mathrm{R}}$) and MCMC effective sample size ($\widehat{\mathrm{ESS}}$) measures. We also describe a method for determining whether an $\widehat{\mathrm{R}}$ diagnostic indicates approximate stationarity of the chains, that may be of more general interest for applications beyond parallel CV. Finally, we show that parallel CV and its diagnostics can be implemented with online algorithms, allowing parallel CV to scale up to very large blocking designs on memory-constrained computing accelerators.

Keywords Bayesian inference · Convergence diagnostics · Parallel computation · $\hat{R}$ statistic

1 Overview

Bayesian cross-validation (CV; Geisser 1975; Vehtari and Lampinen 2002) is a method for assessing models’ predictive ability, and is a popular basis for model selection. Naive or ‘brute force’ approaches to CV, which repeatedly fits models to data subsets, are computationally demanding. Brute force CV is especially costly when the number of model fits is large and inference is performed by Markov chain Monte Carlo (MCMC) sampling. Furthermore, since MCMC inference must be closely supervised to identify issues and to monitor convergence, assessing many models fits can also be labor-intensive. Consequently, brute force CV is often impractical under conventional inference workflows (e.g., Gelman et al. 2020).

Fast alternatives to brute force CV exist for special cases. Importance sampling and Pareto-smoothed importance sampling (Gelfand et al. 1992; Vehtari et al. 2017) require only a single MCMC model fit to approximate leave-one-out (LOO) CV. However, importance sampling is known to fail when the resampling weights have thick-tailed distributions, which is especially likely for CV schemes designed for non-exchangeable data. Examples include hv-block CV for time series applications (Racine 2000) and leave-one-group-out (LOGO) CV for grouped hierarchical models, where several observations are left out at the same time. In these cases, the analyst must usually fall back on brute force methods.

In this paper, we show that general brute force CV by MCMC is feasible on computing accelerator hardware, specifically on graphics processor units (GPUs). Our method, which we call parallel CV (PCV), includes an inference

✉ Alex Cooper
alexander.cooper@monash.edu

Aki Vehtari
Aki.Vehtari@aalto.fi

Catherine Forbes
catherine.forbes@monash.edu

Dan Simpson
dan@normalcomputing.ai

Lauren Kennedy
lauren.a.kennedy@adelaide.edu.au

[1] Department of Econometrics and Business Statistics, Monash University, Clayton, Australia

[2] Department of Computer Science, Aalto University, Espoo, Finland

[3] Normal Computing, New York, USA

[4] School of Computer and Mathematical Sciences, University of Adelaide, Adelaide, Australia

workflow and associated MCMC diagnostic methods. PCV is not a replacement for standard inference workflows, but rather an extension that applies after criticism of candidate models. PCV runs inference for all folds in parallel, potentially requiring thousands of independent MCMC chains, and assesses convergence across all chains simultaneously using diagnostic statistics that target the overall CV objective. Our experiments show that PCV can estimate moderately large CV problems on an 'interactive' timescale—that is, a similar elapsed wall clock time as the original full-data model fit by MCMC.

PCV is an application of massively parallel MCMC, which takes advantage of recent developments in hardware and software (see e.g. Lao et al. 2020) and CV's embarrassingly parallel nature. Unlike the 'short chain' parallel MCMC approach which targets a single posterior, we target multiple independent posteriors concurrently on a single computer. Other approaches include independent chains run on separate CPU cores and/or within-chain parallelism, both of which are available in the Stan language (Stan Development Team 2022). Local balancing approaches parallel inference by generating 'clouds' of proposals for each MCMC step (Glatt-Holtz et al. 2022; Gagnon et al. 2023; Holbrook 2023). Neiswanger et al. (2014) handle large datasets by targeting a single composite posterior with distributed MCMC samplers applied to different data subsets (see also Vehtari et al. 2020b; Scott et al. 2016).

In addition to parallelism, PCV further reduces computational effort compared with naive brute force CV. First, PCV simplifies warm-up runs by reusing information generated during full-data inference, in effect exploiting the similarity between the full-data and partial-data CV posteriors. Second, running chains in parallel allows early termination as soon as the required accuracy is achieved. Since Monte Carlo (MC) uncertainty is usually small relative to the irreducible CV epistemic uncertainty (see Sect. 2.2), applications of CV for model selection typically require only relatively short MCMC runs. The third way is technical, and applies where online algorithms are used. These have small, stable working sets and make effective use of memory caches on modern computer architectures (see e.g. Nissen 2023; Stallings 2015), although we do not analyze this phenomenon in this paper.

Figure 1 previews a PCV model selection application for a hierarchical Gaussian regression model, described in Sect. 2.1. The goal of this exercise is to estimate the probability that one model predicts better than another under the logarithmic scoring rule and a LOGO-CV design. The results clearly stabilize after just a few hundred MCMC iterations. This is explained by the fact that the MC uncertainty is small relative to epistemic uncertainty, so that running the MCMC algorithm for longer confers little additional insight.

In summary, this paper makes the following contributions to a methodological toolkit for parallel CV:

- A practical workflow for fast, general brute force CV on computing accelerators (Sect. 3);
- A massively parallel sampler and associated diagnostics for PCV (Sect. 3.3); and
- A measure of MC and epistemic uncertainty (Sivula et al. 2022), the effective sample size ($\widehat{\mathrm{ESS}}$; Robert and Casella (2004)), and a measure of mixing based on the $\widehat{\mathrm{R}}$ statistic (Gelman and Rubin 1992; Vehtari et al. 2020a) (Sect. 4).
- Examples with accompanying software implementations (Sect. 5 and supplement).

2 Background

This section provides a brief overview of predictive model assessment and MCMC-based Bayesian inference. Consider an observed data vector $y \sim p_{\text{true}}$, where p_{true} denotes some 'correct' but unknown joint data distribution. Suppose an analyst fits some Bayesian model M for the purpose of predicting unseen realizations $\tilde{y}$ from p_{true}. Having fit the posterior distribution $p\left(\theta \mid y, M\right)$, the predictive density f_M is a posterior-weighted mixture,

$$f_{M,y}\left(\tilde{y}\right) = \int p_y\left(\tilde{y} \,|\, \theta, M\right) p\left(\theta \,|\, y, M\right) d\theta. \tag{1}$$

With the predictive $f_{M,y}$ in hand, two natural question arise. First, how well does $f_{M,y}$ predict unseen observations (out of sample)? Second, if multiple models are available, which one predicts better?

2.1 Predictive assessment

If the true data distribution $p_{\text{true}}\left(\tilde{y}\right)$ were somehow known, one could assess the predictive performance of $f_{M,y}$ directly using a scoring rule. A scoring rule $S\left(q, y\right)$ is a functional that maps a predictive density q and a realization y to a numerical assessment of that prediction. A natural summary of the performance of $f_{M,y}$ is the expected score,

$$S_{M,y} = \int p_{\text{true}}\left(\tilde{y}\right) S\left(f_{M,y}, \tilde{y}\right) \mathrm{d}\tilde{y}. \tag{2}$$

It is straightforward to use $S_{M,y}$ as a basis for model selection. For a pairwise model comparison between candidate models, say M_A and M_B, a simple decision rule relies only on the sign of the difference

$$\Delta = S_{M_A,y} - S_{M_B,y}. \tag{3}$$

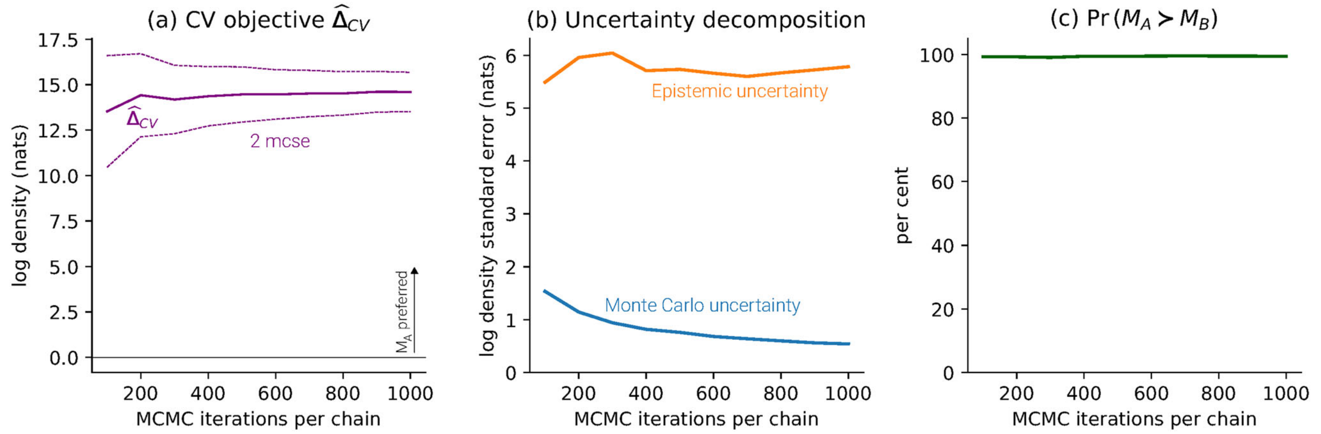

Fig. 1 Model selection for a leave-one-group-out CV (LOGO-CV) for a toy hierarchical Gaussian regression problem, using the log score (Example 1). Model M_A is correctly specified, including all 4 regressors, while M_B is misspecified and includes only 3. The plot shows progressive statistics for MCMC inference using 4 chains per fold, for a total of 200 chains. Panel (**a**) shows the score difference $\widehat{\Delta}$ and 2 Monte Carlo standard error (MCSE) interval as a function of MCMC iterations per chain. Positive (negative) score values favor M_A (M_B). Panel (**b**) shows that epistemic uncertainty dominates MC uncertainty, demonstrating the limited utility of further reducing MC uncertainty. Panel (**c**) shows that Pr $(M_A \succ M_B \mid y)$ quickly stabilizes close to 100%. See Sect. 2.1 for full details

Positive values of Δ indicate M_A is preferred to M_B under S (an event we denote $M_A \succ M_B$) and vice-versa.

Ideally, the choice of scoring rule would be tailored to the application at hand. However, in the absence of an application-driven loss function, generic scoring rules are available. By far the most commonly-used scoring rule is the log predictive density LogS $(f, y) := \log f(y)$, which has the desirable mathematical properties of being local and strictly proper (Gneiting and Raftery 2007). LogS also has deep connections to the statistical concepts of KL divergence and entropy (see, e.g., Dawid and Musio 2014). While we focus on LogS, it is worth noting that LogS has drawbacks too: it requires stronger theoretical conditions to reliably estimate scores using sampling methods and can encounter problems with models fit using improper priors (Dawid and Musio 2015). More stable results can be obtained by alternative scoring rules, albeit at the cost of statistical power (Krüger et al. 2021). To demonstrate the flexibility of PCV in Appendix A we briefly discuss the use of alternative scoring rules, the Dawid-Sebastini score (DSS; Dawid and Sebastiani (1999)) and Hyvärinen score (HS; Hyvärinen and Dayan (2005)).

Throughout the paper, we illustrate ideas using the following example, with results in Fig. 1.

Example 1 (Model selection for a grouped Gaussian regression) Consider the following regression model of grouped data:

$$y_{ij} \mid \alpha_j, \beta, \sigma_y^2 \overset{ind}{\sim} \mathcal{N}(\alpha_j + x_{j[i]}^\top \beta, \sigma_y^2), \tag{4}$$

for $i = 1, \ldots, N_j, \quad j = 1, \ldots, J$. The group effect prior is hierarchical,

$$\alpha_j | \mu_\alpha, \sigma_\alpha^2 \overset{iid}{\sim} \mathcal{N}(\mu_\alpha, \sigma_\alpha^2), \quad j = 1, \ldots, J, \tag{5}$$

where $\mu_\alpha \sim \mathcal{N}(0, 1)$ and $\sigma_\alpha \sim \mathcal{N}^+(0, 10)$, the half-normal distribution with variance 10 and positive support. The remaining priors are $\sigma_y \sim \mathcal{N}^+(0, 10)$ and $\beta \sim \mathcal{N}(0, I)$. Consider two candidate models distinguished by their group-wise explanatory variables x_j. Model M_A is correctly specified, while model M_B is missing one covariate. To assess an observation $\tilde{y}$ from group j' with explanatory variables $\tilde{x}_{j'}$ with respect to LogS, the predictive density is

$$f_{M,y}(\tilde{y}) = \int \mathcal{N}\left(\tilde{y} \mid \alpha_{j'} + \tilde{x}_{j'}^\top \beta, \sigma_y^2\right) p\left(\alpha_{j'} \mid y, M\right) p\left(\beta, \sigma_y^2 \mid y, M\right) d\alpha_{j'}\, d\beta\, d\sigma_y^2. \tag{6}$$

Here and for the remainder of the paper, $\mathcal{N}\left(x \mid \mu, \sigma^2\right)$ denotes the density of $\mathcal{N}\left(\mu, \sigma^2\right)$ evaluated at x. In the case where group j' does not appear in the training data vector y, the marginal posterior density $p\left(\alpha_{j'} \mid y, M\right)$ that appears in (6) is defined using the posterior distributions for μ_α and σ_α, as

$$\int p\left(\alpha_{j'} | \mu_\alpha, \sigma_\alpha\right) p\left(\mu_\alpha, \sigma_\alpha \mid y, M\right) d\mu_\alpha d\sigma_\alpha. \tag{7}$$

We simulate $J = 50$ groups with $N_j = 5$ observations per group. The elements of the $J \times 4$ matrix X is simulated using $\mathcal{N}(0, 10)$ variates. The true values of α, β and σ^2 used in the simulation are drawn from the priors. We use $L = 4$ chains and $K = J = 50$ folds, a total of 200 chains.

2.2 Cross-validation

In practice the true process $p_{\text{true}}(\tilde{y})$ is not known, so (2) cannot be computed directly. Rather, CV approximates (2) using only the observed data y instead of future data $\tilde{y}$. CV proceeds by repeatedly fitting models to data subsets, assessing the resulting model predictions on left-out data. A CV scheme includes the choice of scoring rule and a data partitioning scheme that divides the data into pairs of mutually disjoint test and training sets $(\text{test}_k, \text{train}_k)$, for $k = 1, \ldots, K$. The resulting partial-data posteriors $p\left(\cdot \mid y_{\text{train}_k}, M\right)$ can be viewed as K random perturbations of the full-data posterior $p(\cdot \mid y, M)$.

Popular CV schemes include leave-one-out (LOO-CV) which drops a single observation each fold, and K-fold which divides the data into K disjoint subsets. However, for non-exchangeable data, CV schemes often need to be tailored to the underlying structure of the data, or to the question at hand. For example, time-series, spatial, and spatio-temporal applications can benefit from specifically tailored partitioning schemes (see e.g., Roberts et al. 2017; Cooper et al. 2024; Mahoney et al. 2023). For some data structures the resulting K can be particularly large, such as for LOO-CV and $h(v)$-block CV.

There may be several appropriate CV schemes for a given candidate model and dataset. The CV scheme should also reflect the nature of the generalization required. For example, in hierarchical models with group effects (Example 1), LOGO-CV measures the model's ability to generalize to unseen data groups, while non-groupwise schemes applied to the same model and data would characterize predictive ability for the current set of groups only.

We focus on two CV objectives. First, the CV score $\widehat{S}_M$ is an estimate of predictive ability S_M. It is constructed as a sum over all K folds,

$$\widehat{S}_M = \sum_{k=1}^{K} S\left(f_{M,k}, y_{\text{test}_k}\right), \tag{8}$$

where $f_{M,k}$ denotes the model M predictive constructed using the posterior $p\left(\theta \mid y_{\text{train}_k}\right)$. The quantity $\widehat{S}_M$ can be viewed as a Monte Carlo estimate of S_M, up to a scaling factor. Second, a CV estimate for the model selection objective Δ in (3) is the sum of the K differences,

$$\begin{aligned}\widehat{\Delta} &:= \widehat{S}_{M_A} - \widehat{S}_{M_B} \\ &= \sum_{k=1}^{K} \left[S\left(f_{M_A,k}, y_{\text{test}_k}\right) - S\left(f_{M_B,k}, y_{\text{test}_k}\right)\right] \\ &=: \sum_{k=1}^{K} \widehat{\Delta}_k.\end{aligned} \tag{9}$$

Herein we use the generic notation $\widehat{\eta} = \sum_{k=1}^{K} \widehat{\eta}_k$ to denote the CV objective, whether it be $\widehat{S}_M$ or $\widehat{\Delta}$.

2.3 Epistemic uncertainty

The MC estimators in (8) and (9) are random quantities that are subject to sampling variability. The associated predictive assessments of out-of-sample model performance are therefore subject to uncertainty (Sivula et al. 2022). Naturally, there would be no uncertainty at all if p_{true} were fully known or if an infinitely large dataset were available. But since (8) and (9) are estimated from finite data, an assessment of the uncertainty of these predictions is useful for interpreting CV results (Sivula et al. 2022, Section 2.2). We call this random error *epistemic uncertainty*.

It can be helpful to view as-yet-unseen data as missing data. CV imputes that missing data with finite number of left-out observations. The epistemic uncertainty about the unseen future data can be regarded as uncertainty from the imputation process.

For a given dataset, epistemic uncertainty is irreducible in the sense that it cannot be driven to zero by additional computational effort or the use of more accurate inference methods. This contrasts with Monte Carlo uncertainty in MCMC inference, discussed in the next subsection. The limiting factor is the information in the available dataset.

We adopt the following popular approach to modeling epistemic uncertainty. First, we regard the individual contributions $(\widehat{\eta}_k)$ to $\widehat{\eta}$ as exchangeable, drawn from a large population with finite variance (Vehtari and Lampinen 2002; Sivula et al. 2022). That is, we presume that K is reasonably large. Then, $\widehat{\eta}$ will satisfy a central limit theorem (CLT) so that a normal approximation for the out-of-sample predictive performance of $\widehat{\eta}$ is appropriate.

In particular, for model selection applications we are interested in the probability that M_A predicts better than M_B, which we denote $\Pr(M_A \succ M_B \mid y)$. We approximate

$$\Pr(M_A \succ M_B \mid y) \approx \Phi\left(\frac{\widehat{\Delta}}{\sqrt{K\widehat{\sigma}^2_{\widehat{\Delta}}}}\right), \tag{10}$$

where $\widehat{\sigma}^2_{\widehat{\Delta}} = \frac{1}{K-1}\sum_{k=1}^{K}\left(\widehat{\Delta}_k - \widehat{\Delta}/K\right)^2$ is the sample variance of the contributions to (9) and Φ denotes the standard normal cdf.

There are other ways to model $\Pr(M_A \succ M_B \mid y)$, such as the Bayesian Bootstrap (Rubin 1981; Sivula et al. 2022). We prefer the normal approximation in this setting because it performs well (Sivula et al. 2022) and is simple to approximate with online estimators, making it particularly useful on parallel hardware.

2.4 MCMC inference

MCMC is by far the dominant method for conducting Bayesian inference. It characterizes posterior distributions with samples from a stationary Markov chain, for which the invariant distribution is the target posterior. MCMC sampling algorithms are initialized with a starting parameter value $\theta^{(0)}$, usually in a region of relatively high posterior density. Then, an MCMC algorithm is used to sequentially draw a sample $\theta^{(1)}, \theta^{(2)}, \ldots$ from the chain. This sample can be used to construct Monte Carlo estimators for functionals $g(\theta)$ such as the predictive density $f_{M,y}(\tilde{y})$ that appears in (1). Many MCMC algorithms are available: Gelman et al. (2014) provide a general overview.

For brute-force CV applications, posteriors are separately estimated for each model and fold. Furthermore, typical inference workflows for MCMC inference call for draws from $L > 1$ independent chains targeting the same posterior. For model M, fold k, and chain ℓ, denote the sequence of N MCMC parameter draws $\theta^{(0)}_{M,k,\ell}, \theta^{(1)}_{M,k,\ell}, \ldots, \theta^{(N)}_{M,k,\ell}$, so that the expectation $\mathbb{E}\left[g(\theta) \mid M, y_{\mathsf{train}_k}\right]$ can be estimated for each model M and fold k as

$$\widehat{g}^{(N)}_{M,k} = \frac{1}{LN}\sum_{\ell=1}^{L}\sum_{n=1}^{N} g\left(\theta^{(n)}_{M,k,\ell}\right). \tag{11}$$

The main assumption needed is quite standard (see e.g. Jones et al. (2006)): a CLT for $\widehat{g}^{(N)}_{M,k}$, so that as $N \to \infty$,

$$\sqrt{N}\left(\widehat{g}^{(N)}_{M,k} - \mathbb{E}\left[g(\theta) \mid M, y_{\mathsf{train}_k}\right]\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{\sigma^2_{\widehat{g}_{M,k}}}{L}\right). \tag{12}$$

Because MCMC parameter draws are auto-correlated, a naive estimate of the uncertainty associated with (11) from these draws will be biased. Instead, a Monte Carlo standard error (MCSE) estimator should be used. We use MCSE estimators based on batch means (Jones et al. 2006) because these can be efficiently implemented on accelerator hardware (see Appendix A). Let the chain length be a whole number representing a batches each of b samples, so that $N = ab$. Then the hth batch mean is given by

$$\bar{g}^{(h)}_{M,k,\ell} = \frac{1}{b}\sum_{n=(h-1)b+1}^{hb} g\left(\theta^{(n)}_{M,k,\ell}\right). \tag{13}$$

The MCSE $\sigma_{\hat{g}_{M,k}}/\sqrt{LN}$ can then be computed using the sample variance of the $\bar{g}^{(h)}_{M,k,\ell}$s across all L chains for model M and fold k, where

$$\hat{\sigma}^2_{\widehat{g}_{M,k}} = \frac{b}{La-1}\sum_{\ell=1}^{L}\sum_{h=1}^{a}\left(\bar{g}^{(h)}_{M,k,\ell} - \widehat{g}^{(N)}_{M,k}\right)^2. \tag{14}$$

There is a large literature on estimators for $\sigma^2_{\widehat{g}_{M,k}}$, but we use (14) since it is simple and it performed well in our experiments. The batch size b is a hyper-parameter to be chosen before inference starts, and large enough for the Y_k to be approximately independent. Where the MCMC chain length is known (in the case where a data-dependent stopping rule is not used), asymptotic arguments suggest $b = \left\lfloor\sqrt{NL}\right\rfloor$, for which a rough guess can be made *a priori* (see for instance Jones et al. (2006) for a discussion).

To reliably fit Bayesian models, the inference workflow needs to include careful verification of model fit. MCMC algorithms must also be carefully checked for pathological behaviors and monitored for convergence so that inference can be terminated (Gelman et al. 2020). Most workflows are oriented toward parameter inference, ensuring that the samples adequately characterize the desired posterior distribution $p(\theta \mid y)$. Assessing convergence effectively amounts to verifying that (a) each posterior's chains are correctly mixing, and (b) the sample size is large enough to characterize the posterior distribution to the desired degree of accuracy. To support these assessments, several diagnostic statistics are available (Roy 2020). We describe two diagnostic statistics specifically adapted for parallel MCMC in Sect. 4.

3 Parallel cross-validation

In this section we describe the proposed PCV workflow. PCV aims to make brute force CV feasible by reducing computation (wall clock) time as well as the analyst's time spent checking diagnostics. Of course, given unlimited computing power and effort, an analyst could simply implement Bayesian CV by sequentially fitting each CV fold using MCMC, checking convergence statistics for each, then constructing the objective $\widehat{\eta}$ using (8) or (9). This is computationally and practically infeasible for CV designs with large K.

Many existing applications of massively parallel MCMC target just a single posterior. In contrast, PCV simultaneously estimates multiple posteriors using parallel samplers that execute in lock-step. Under our approach, the algorithm draws vectors of combined parameter vectors representing all chains for all folds. Estimating $\hat{\eta}$ by brute force MCMC requires samples from $\mathcal{O}(KL)$ parallel independent MCMC chains. In some cases, posteriors for different models can be sampled in parallel, too (see Appendix B).

The main challenge that must be solved for estimating all K posteriors (or $2K$ posteriors for a pairwise model comparison) is that model criticism, diagnostics, and convergence checking must be applied to all of the posteriors. However, since full model criticism and checking should be performed on the full-data models anyway, for which the partial-data posteriors are simply random perturbations, we argue that model criticism need only be applied to the full-data candidate models. Furthermore, we can sharply reduce the computational effort required for sampling by using information from full-data inference to initialize the partial-data inference (Sect. 3.1).

We propose the following extension to conventional Bayesian inference workflows:

Step 1. Full-data model criticism and inference. Perform model criticism on candidate model(s) using the full data set (Gelman et al. 2020), revising candidate models as necessary, and obtain MCMC posterior draws for each model;

Step 2. Parallel MCMC warmup (see Sect. 3.1). Initialize L parallel chains for each of K folds using random draws from the full-data MCMC draws obtained in Step 1. Run short warm-up chains in parallel and discard the output;

Step 3. Parallel sampling. Run the $\mathcal{O}(KL)$ parallel MCMC chains for N iterations, accumulating statistics required to evaluate $\widehat{\eta}$ and associated uncertainty measures (see Sect. 3.2); and

Step 4. Check convergence. Compute and check parallel inference diagnostics (Sect. 4), and if necessary adjust inference settings and repeat.

3.1 Efficient MCMC warmup

General-purpose MCMC sampling procedures typically begin with a warmup phase. The warmup serves two goals: (i) it reduces MCMC estimator bias due to initialization, and (ii) adapts tuning parameters of the sampler. Warm-up procedures aim to ensure the distribution of the initial chain values $\theta_\ell^{(0)}$ is close to that of the target posterior. An example is Stan's window adaptation algorithm (Stan Development Team 2022), designed for samplers from the Hamiltonian Monte Carlo (HMC; Neal 2011) family. Hyperparameter choices are algorithm-specific: for example, HMC requires a step size, trajectory length, and inverse mass matrix.

The warm-up phase is computationally costly. It would be especially costly and time consuming to run complete, independent tuning procedures for each fold in parallel to obtain kernel hyperparameters and initial conditions $\mathcal{O}(KL)$ for each fold. In addition, running many independent tuning procedures can be unreliable. Tuning procedures are stochastic in nature, and as the number of chains increases, the probability of initializing at least one chain with problematic starting conditions increases. An example of such a starting condition is a parameter draw far in a region of the parameter space that leads to numerical problems and a 'stuck chain'.

Instead of running independent warm-up procedures for each fold, we propose re-using the warm-up results from the full-data model for each CV fold, under the assumption that the full-data and partial-data posteriors are close. This assumption seems reasonable if the folds are similar enough to the full-data model for CV to be interpretable as a predictive assessment of the full-data model (Sect. 2.2).

Under this approach, each fold's MCMC kernel uses the same inference tuning parameters (e.g. step size and trajectory length) as the full-data model. Starting positions are randomly drawn from the full-data posterior MCMC sample. To ensure distribution of the starting conditions are close to the fold model's posterior distribution, PCV then simulates and discards a very short warm-up sample.

3.2 Estimating uncertainty

Practical CV applications require estimates of the uncertainty of $\widehat{\eta}$ estimates. Both MC and epistemic uncertainty contribute to the variability in $\widehat{\eta}$.

We estimate epistemic uncertainty by applying the normal approximation described in Sect. 2.3. For model selection application, we substitute fold-level estimates $\left(\widehat{\Delta}_k\right)$ into (10).

MC uncertainty can be estimated using an extension of standard methods described in Sect. 2.4. Because each fold is estimated using independent MCMC runs, the overall MC uncertainty for $\widehat{\eta}$ is simply the sum of the MC variance of each fold's contribution. When an estimated scoring rule may be represented as a smooth function of an ergodic mean (like LogS), $\sigma^2_{\widehat{\eta}}$ can be estimated using the delta method. In the case of LogS, we have

$$\sigma^2_{\widehat{\eta}} = \sum_{M\in\mathcal{M}}\sum_{k=1}^{K}\sigma^2_{S_{M,k}} \approx \sum_{M\in\mathcal{M}}\sum_{k=1}^{K}\frac{\sigma^2_{\hat{f}_{M,k}}}{\left(\hat{f}_{M,k}\right)^2}, \tag{15}$$

where summation over models reflects the fact that the MC error for a difference is the sum of the error for both terms. The MCSE for $\hat{\eta}$ is then $\mathrm{MCSE}_{\hat{\eta}} = \sigma_{\hat{\eta}}/\sqrt{LN}$. Independence of the contributions to the overall MC error is helpful for producing accurate estimates quickly.

Even when η and its associated standard error can theoretically be estimated using MC estimators (for instance if its first two moments are finite), in practice theoretical conditions may not be enough to prevent numerical problems during inference. A common cause of numerical overflow is the presence of outliers that fall far in the tails of predictive distributions.

In model selection applications, it is typical for MC uncertainty to be an order of magnitude smaller than epistemic uncertainty (see e.g. Fig. 1). This discrepancy implies that, provided that the chains have mixed, long MCMC runs are not usually necessary in model selection applications, since additional effort applied to MCMC sampling will not meaningfully improve the accuracy of $\widehat{\eta}$. An overall measure that provides a single picture of uncertainty is also useful because the MCMC efficiency of individual folds can vary tremendously (see Sect. 4.1 and Fig. 2).

Since PCV applications typically require only relatively small N to make MC uncertainty insignificant compared to epistemic uncertainty, we do not propose a stopping rule for of the type discussed by Jones et al. (2006). In most cases these rules are justified by asymptotic arguments (i.e. for large N), whereas in our examples N is on the order of 10^3.

3.3 Implementation on accelerator hardware

Massively parallel samplers are able to take advantage of modern computing accelerators, which can offer thousands of compute units per device. Accelerators typically offer more cost-effective throughput, measured in floating-point operations per second (FLOPS), than conventional CPU-based workstations. However, despite their impressive parallel computing throughput and cost per FLOPS, the design of computing accelerators impose heavy restrictions on the design of inference algorithms.

Programs with heavy control flow beyond standard linear algebra operations tend to be inefficient, including those commonly used in Bayesian inference such as sorting large data vectors. In addition, accelerators have limited onboard memory for storing and manipulating draws generated by MCMC samplers. The need to transfer MCMC draws to main memory for diagnostics and manipulation would represent a significant performance penalty. These problems could be alleviated if all analysis steps could be conducted on the accelerator device, within its memory limits.

The Hamiltonian Monte Carlo (HMC; Neal 2011) sampling algorithm lends itself well to implementation on accelerator hardware. To implement PCV, we augment the HMC kernel with an additional parameter representing the fold identifier. This allows the unnormalized log joint density—and its gradient via automatic differentiation—to select the appropriate data subsets for each chain. HMC is suitable for parallel operations because its integrator follows a fixed trajectory length at each MCMC iteration. The lack of a dynamic trajectory allows HMC to be vectorized across a large number of parallel chains, with each evolving in lockstep. In contrast, efficient parallel sampling is extremely difficult with dynamic algorithms such as the popular No-U-Turn Sampler (NUTS; Hoffman et al. 2014). Assessing chain efficiency (Sect. 4.1) is more important with samplers with non-adaptive step size like HMC, since samples tend to be more strongly auto-correlated than for adaptive methods, delivering fewer effective draws per iteration.

A further constraint inherent to computing accelerators is the size of the device's on-board memory. Limited accelerator memory means it is usually infeasible to store all MCMC parameter draws from all chains for later analysis, which requires only $\mathcal{O}\left(KLN\dim(\theta)\right)$ memory. This cost can be prohibitive, especially when $\dim(\theta)$ is large. However, in many cases it is feasible to store draws required to construct the objective, that is the univariate draws required to estimate $\hat{\eta}$, reducing the memory requirement to $\mathcal{O}(KLN)$. This approach is very simple to implement (see Algorithm 1).

Where accelerator device memory is so constrained that even the draws for $\hat{\eta}$ cannot be stored on the accelerator, online algorithms are available. The memory footprint for such algorithms does not depend on the chain length N. However, as Algorithm A2 in the supplementary material demonstrates, the fully-online approach is significantly more complicated. (See also the sampler implemented in Appendix D in the supplementary material.) While the online approach is very memory efficient, it is also less flexible. Conventional workflows recommend first running inference then computing diagnostics from the resulting draws. However, under the online approach one must predetermine which diagnostics will be run after inference, then accumulate enough information during inference to compute those diagnostics without reference to the full set of predictive draws (see Appendix A).

Algorithm 1 Non-online parallel CV sampler for LogS. Superscripts index array elements. See also Algorithm A2 in the supplementary material.

```
Input: Posterior draws $\left\{\theta_1^{(\mathrm{fd})}, \ldots, \theta_{N_{\mathrm{fd}}}^{(\mathrm{fd})}\right\}$, MCMC kernels
  $\{\mathcal{T}_k(\theta, \cdot)\}_{k=1}^{K}$, log predictive densities $\{\log p(\tilde{y}_k \mid \theta)\}_{k=1}^{K}$, folds
  $K$, chains $L$, warm-up length $N_{\mathrm{wu}}$, chain length $N$
Output: $\hat{s}$.
  Initialize:
    $K \times L \times N$ array $\hat{s}$
  for $k \in 1, \ldots, K$ do in parallel:                          ▷ fold loop
    for $\ell \in 1, \ldots, L$ do in parallel:                     ▷ chain loop
      $\theta^{(k,\ell)} \leftarrow$ draw from $\left\{\theta_1^{(\mathrm{fd})}, \ldots, \theta_{N_{\mathrm{fd}}}^{(\mathrm{fd})}\right\}$
      for $i \in 1, \ldots, N_{\mathrm{wu}}$ do sequentially   ▷ warm-up sampling loop
        $\theta^{(k,\ell)} \leftarrow$ draw from $\mathcal{T}_k\left(\theta^{(k,\ell)}, \cdot\right)$
      end for
      for $i \in 1, \ldots, N$ do sequentially                     ▷ sampling loop
        $\theta^{(k,\ell)} \leftarrow$ draw from $\mathcal{T}_k\left(\theta^{(k,\ell)}, \cdot\right)$
        $\hat{s}^{(k,\ell,i)} \leftarrow \log p\left(\tilde{y}_k \mid \theta^{(k,\ell)}\right)$
      end for
    end for
  end for
```

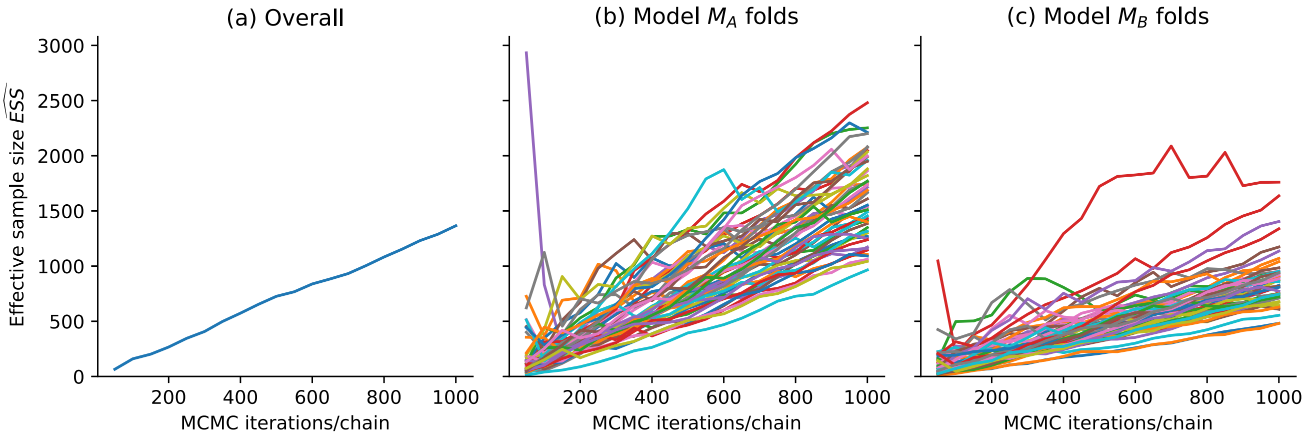

Fig. 2 Progressive estimated effective sample size ($\widehat{\text{ESS}}$) for Example 1 (grouped Gaussian regression), a scale-free measure of information content for the MCMC sample, as a function of MCMC iterations. Panel (**a**) shows $\widehat{\text{ESS}}$ for the overall model selection statistic (incorporating all chains), while panels (**b**) and (**c**) respectively show the $\widehat{\text{ESS}}$ for each fold of models M_A and M_B. Note the overall measure falls within the range of all folds

4 MCMC diagnostics

We propose that diagnostics focus on $\widehat{\eta}$ rather than fold-specific parameters. Under our proposed workflow, the analyst will have completed model criticism on the full-data model before attempting brute force CV. Soundness of the full-data model strongly suggests that the CV folds, which by construction have the same model structure and similar data, will also behave well. However, inference of all folds should nonetheless be monitored to ensure convergence has been reached, and to identify common problems that may arise during computation.

A focus on the predictive quantity $\hat{\eta}$ rather than the model parameters carries several advantages. First, it provides a single view of convergence that targets the desired output and ignores any inference problems in irrelevant parts of the model, such as group-level random effects that are not required to predict the group of interest. Second, unlike parameter convergence diagnostics, diagnostics for $\hat{\eta}$ are sensitive to numerical issues arising in the predictive components of the model. Third, these diagnostics can be significantly cheaper to compute than whole-parameter diagnostics, in part because the target is univariate or low-dimensional.

Other diagnostic statistics for massively parallel inference include $\mathfrak{n}\widehat{R}$ (Margossian et al. 2023), which is applicable to large numbers of short chains targeting the same posterior. In contrast, our diagnostics target a small number of longer chains per fold, and are applied to a large number of distinct posteriors.

4.1 Effective sample size

An (estimated) effective sample size ($\widehat{\text{ESS}}$; Geyer (1992)) provides a scale-free measure of the information content of an autocorrelated sample. Since MCMC samples from a single chain are not independent, an estimate of the limiting variance σ_g^2 in (12), denoted by $\widehat{\sigma}_g^2$, is typically greater than the usual sample variance s_g^2, and hence the degree of autocorrelation must be taken into account when computing the Monte Carlo standard error (MCSE) of estimates computed from the resulting MCMC sample.

The standard ESS measure targets individual parameter estimates, say θ_i. Define $\widehat{\text{ESS}}_{\theta_i}$, as the raw sample size adjusted by the ratio of the unadjusted sample variance $s_{\theta_i}^2$ to the corresponding MC variance $\hat{\sigma}_{\theta_i}^2$: $\widehat{\text{ESS}}_{\theta_i} = LNs_{\theta_i}^2/\hat{\sigma}_{\theta_i}^2$. For parallel inference we use the batch means method described in Sect. 2.4 to estimate $\hat{\sigma}_{\theta_i}^2$, for which online estimators are simple to implement. (For alternatives see e.g., the review by Roy (2020).)

For PCV, an aggregate measure of the sample size $\widehat{\text{ESS}}$ can be computed similarly. Define $\widehat{\text{ESS}} = LNs_{\hat{\eta}}^2/\hat{\sigma}_{\hat{\eta}}^2$. $\widehat{\text{ESS}}$ is useful as a single scale-free measure across all folds (see Fig. 2).

4.2 Mixing: aggregate $\widehat{R}_{\text{max}}$

To assess mixing of the ensemble of chains, we propose a combined measure of mixing based on the potential scale reduction factor $\widehat{\text{R}}$ (Gelman and Rubin 1992; Vehtari et al. 2020a). Most of the chains in the ensemble should have $\widehat{R}$s below a suitable threshold (Vehtari et al. (2020a) suggest 1.01). In addition, we assess overall chain convergence by $\widehat{R}_{\text{max}}$, the maximum of the $\widehat{\text{R}}$s measured across all folds.

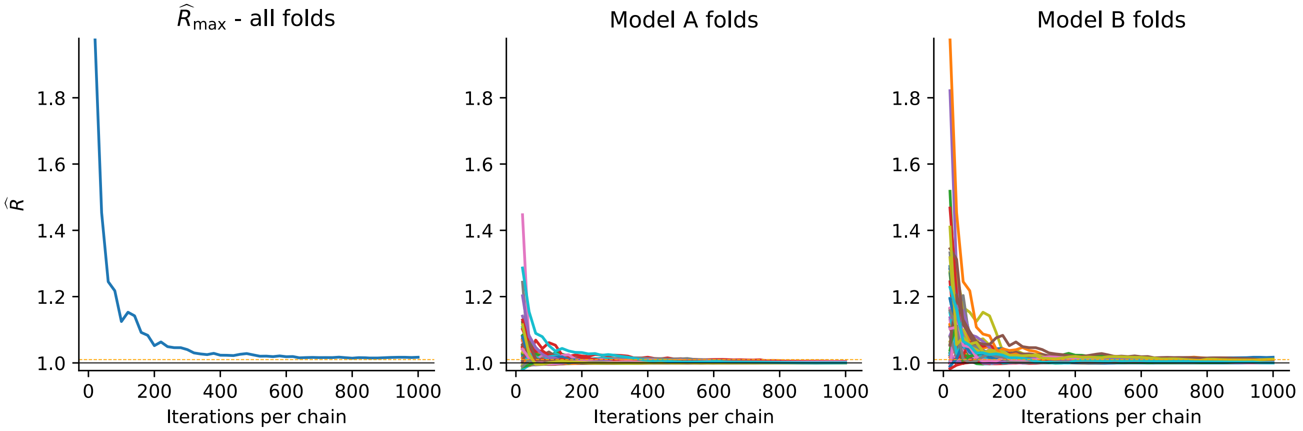

Fig. 3 Progressive $\widehat{\mathrm{R}}$ measures as a function of iterations/chain for the toy grouped regression model comparison. Note that the 1.01 threshold (Vehtari et al. 2020a) is exceeded by at least one chain, and hence by $\widehat{R}_{\max}$, for most of the 1000 iterations/chain plotted in this diagram

Below, we describe a simple problem-specific method for interpreting $\widehat{R}_{\max}$.

The canonical $\widehat{\mathrm{R}}$ measure aims to assess whether the independent chains targeting the same posterior adequately characterize the whole posterior distribution. $\widehat{R}$ usually targets parameter means, but in our experiments we found that the mean of the log score draws to be a useful $\widehat{R}$ target. Other targets are of course possible, and wide range of other functionals appear in the literature (e.g. Vehtari et al. 2020a; Moins et al. 2023).

$\widehat{\mathrm{R}}$ is a scaled measure of the variance of between-chain means B, a quantity that should decrease to zero as the chains converge in distribution and become more similar. Several variants of $\widehat{\mathrm{R}}$ exist. To simplify computation on accelerators, the simple version we use here omits chain splitting and rank-normalization (these features are described by Vehtari et al. (2020a)). For a given model M and fold k, define $\widehat{\mathrm{R}}$ as

$$\widehat{\mathrm{R}}_{M,k} = \sqrt{\frac{\frac{N-1}{N}\widehat{W}_{M,k} + \frac{1}{N}\widehat{B}_{M,k}}{\widehat{W}_{M,k}}}. \tag{16}$$

The within-chain variance $W_{M,k}$ and between-chain variance $B_{M,k}$ are, respectively

$$W_{M,k} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{1}{N-1}\sum_{n=1}^{N}\left(\hat{s}_{M,k,\ell}^{(n)} - \bar{s}_{M,k,\ell}\right)^2,$$

$$B_{M,k} = \frac{N}{L-1}\sum_{\ell=1}^{L}\left(\bar{s}_{M,k,\ell} - \bar{s}_{M,k,\cdot}\right)^2,$$

where $\hat{s}_{M,k,\ell}^{(n)}$ is the nth draw of $\log p(\tilde{y}|\theta)$ in chain ℓ, $\bar{s}_{M,k,\ell}$ is the chain sample mean, and $\bar{s}_{M,k,\cdot}$ is the sample mean of parameter draws for the fold k chains.

The summary mixing measure is then

$$\widehat{R}_{\max} = \max_{M\in\mathcal{M},k=1,\ldots,K} \widehat{R}_{M,k}. \tag{17}$$

Since all the $\widehat{\mathrm{R}}_{M,k}$ tend to 1 as chains converge and K is fixed, it follows that $\widehat{R}_{\max}$ tends to 1 as all posterior chains converge.

However, while $\hat{R}_{\max}$ has the same limiting value as $\widehat{\mathrm{R}}$, it is not at all clear that the broadly accepted threshold of $\widehat{\mathrm{R}} < 1.01$ (Vehtari et al. 2020a) for a single posterior is an appropriate indicator that all folds have fully mixed. Each $\widehat{\mathrm{R}}_{M,k}$ is a stochastic quantity, and the extremum statistic $\widehat{R}_{\max}$ is likely to be large relative to the majority of chains.

Figure 3 compares $\widehat{R}_{\max}$ with the $\widehat{\mathrm{R}}$ computed for each fold of both regression models M_A and M_B. Recall from Fig. 1 that that the $\widehat{\eta}$ estimates stabilize after a few hundred MCMC iterations per chain. However, well beyond this point, $\widehat{R}_{\max}$ exceeds the conventional convergence threshold for $\widehat{\mathrm{R}}$ of 1.01 (Vehtari et al. 2020a).

To estimate an appropriate benchmark for $\widehat{R}_{\max}$ for a given problem, we propose the following simulation-based procedure. This procedure empirically accounts for the autocorrelation in each chain, without the need to model the behavior of each fold's posterior, and with only minimal additional computation. This approach is conceptually similar to the block bootstrap, and it directly accounts for the autocorrelation in each fold's MCMC chains.

Suppose (hypothetically) that all chains are well-mixed, so that the mean and variance of any chain should be roughly the same. In that case, if we also assume that autocorrelation is close to zero within the block size, then computing $\widehat{\mathrm{R}}$ should not be greatly affected if blocks of each chains are 'shuffled' as shown in Panel (a) of Fig. 4. To construct an estimate of the likely range of $\widehat{\mathrm{R}}$ values under the assumption that the chains have mixed, we simply repeatedly compute $\widehat{\mathrm{R}}$ from a large

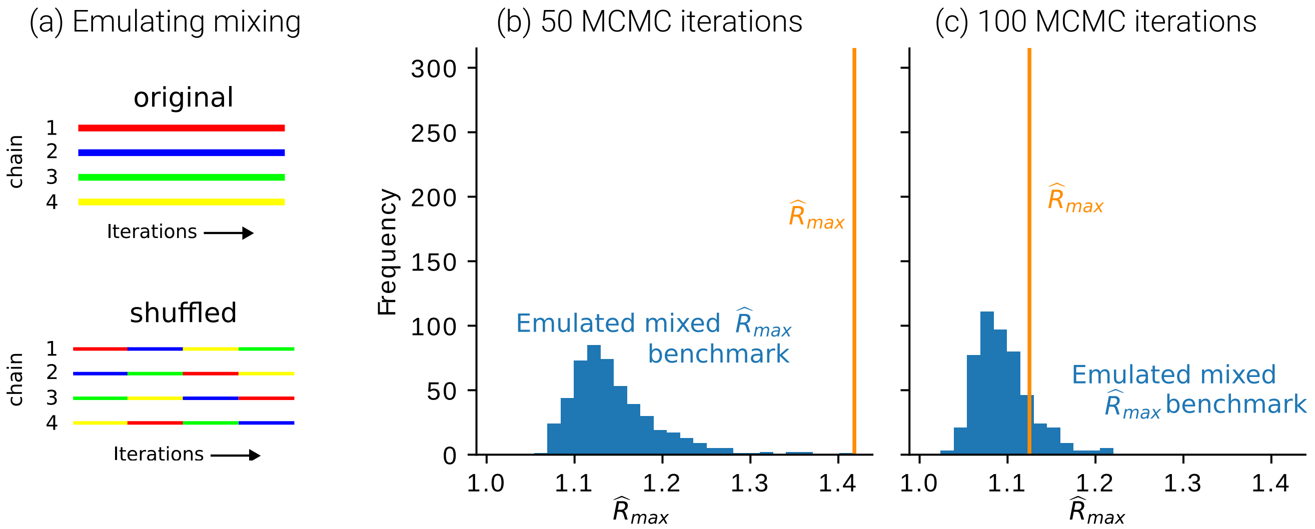

Fig. 4 Two values of $\widehat{R}_{\max}$ for the toy regression problem, compared with emulated mixed $\widehat{R}_{\max}$ benchmark draws, computed by block-shuffling chains. Panel (**a**) shows a stylized shuffling scheme, where chains are broken into contiguous blocks and recombined by shuffling with replacement. Panels (**b**) and (**c**) show $\widehat{R}_{\max}$ estimates at 50 and 100 parallel MCMC iterations, respectively (vertical line), alongside histograms of 500 shuffled $\widehat{R}_{\max}$ draws for comparison. 5 blocks were used. In this example, we conclude that the parallel chains have not converged after 50 iterations, but they have after 100

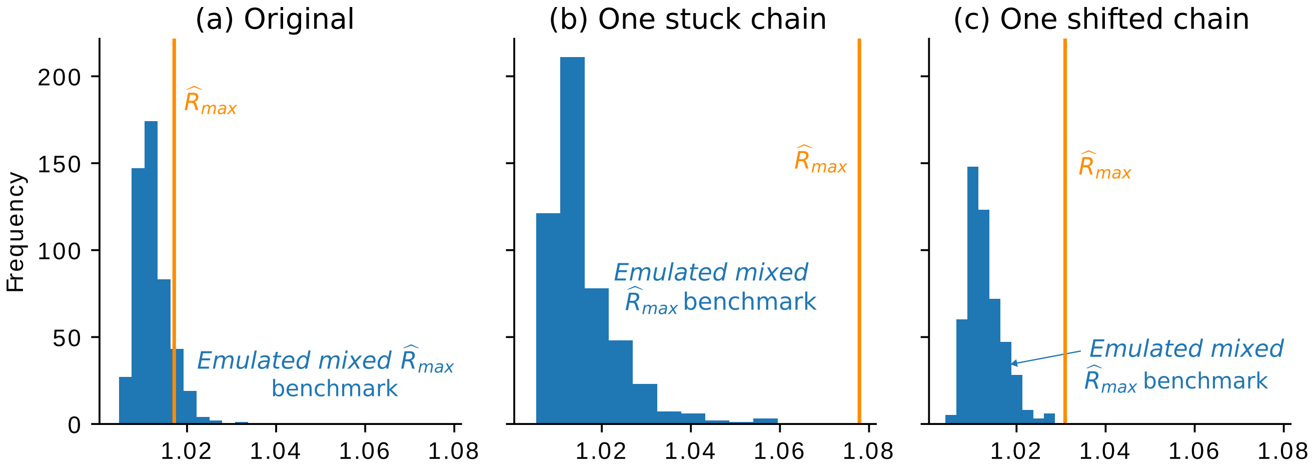

Fig. 5 Artificially constructed pathological inference examples, detected by $\widehat{R}_{\max}$. Panel (**a**) shows the original comparison in Fig. 4, after 1000 iterations. In Panel (**b**) a single chain for one of the folds has been fixed to a constant value (its first draw). In Panel (**c**) a single chain for one of the folds has been shifted by 5 units. In both panels (**b**) and (**c**), $\widehat{R}_{\max}$ lies to the right of the histogram of 100 emulated stationary $\widehat{R}_{\max}$ draws, estimated using 5 blocks

sample of shuffled draws. Rather than adopt a threshold based on an arbitrary summary statistic of the shuffled draws as a single benchmark (say an upper quantile), we instead simply present the draws as a histogram for visual comparison.

Figure 5 demonstrates $\widehat{R}_{\max}$ detecting two artificially-created pathological conditions: a stuck chain and a shifted chain, both of which correspond to non-convergence of one of the folds. To be clear, we do not claim that this $\widehat{R}_{\max}$ benchmark is foolproof or even that it will detect most convergence issues, but it did perform well in our examples when the bulk of folds had mixed (Sect. 5).

5 Illustrative examples

In this section we present three additional applied examples of PCV. Each example uses PCV to select between two candidate models for a given application. Code can be found online at https://github.com/kuperov/ParallelCV. For each

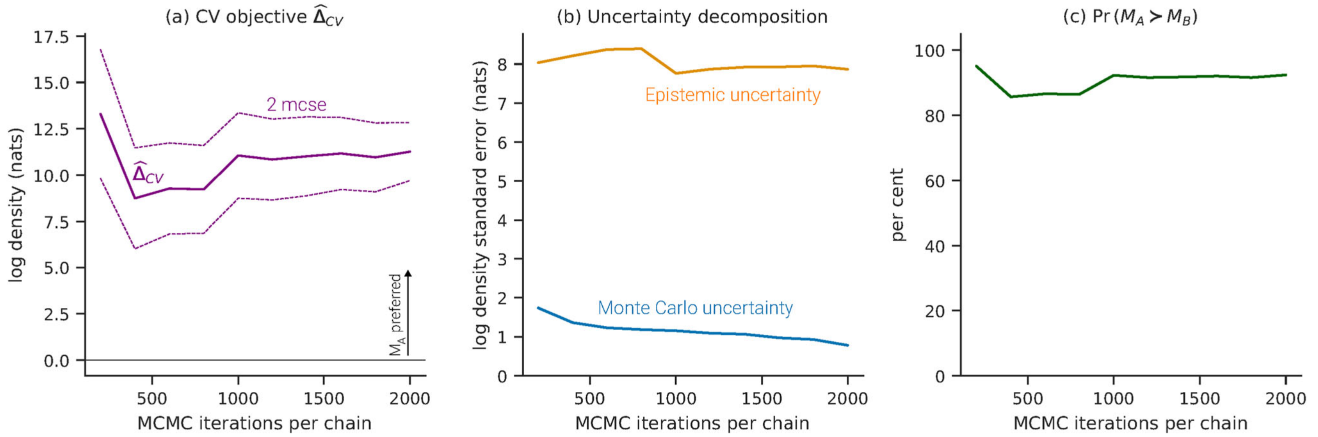

Fig. 6 Progressive estimates for groupwise cross-validation for Example 2 (rat weight). The model selection statistic $\widehat{\Delta}$ shown in Panel (**a**) is clearly positive, favoring model M_A. Panel (**b**) shows that MC uncertainty is a very small component of the total. Panel (**c**) shows that the probability of M_A predicting better than M_B stabilizes after about 1000 iterations

candidate model, we first perform model checking on the full-data model prior to running CV, where chains are initialized using prior draws. For all experiments, we fix the batch size $b = 50$, and check that this batch size yields reasonable ESS estimates compared with window methods (Kumar et al. 2019) on the full-data models.

The core procedure we use for our experiments requires only a log joint likelihood function and log predictive density function compatible with JAX's primitives, and is therefore amenable to automatic vectorization. Our examples use the HMC and window adaptation implementations in Blackjax v1.0 (Lao and Louf 2020), as well as primitives in TensorFlow Probability (TFP; Dillon et al. (2017)). All experiments use double-precision (64-bit) arithmetic. Full-data inference is performed on the CPU while parallel inference is run on the GPU. CPU and GPU details are noted in each results table.

Example 2 (Rat weight) This example demonstrates PCV on grouped data. Gelfand et al. (1990) present a model of the weight of $J = 30$ rats, for each of which five weight measurements are available. The rat weights are modeled as a function of time,

$$M_A : y_{j,t}|\alpha_j, \beta_j, \sigma_y \sim \mathcal{N}\left(\alpha_j + \beta_j t, \sigma_y^2\right), \tag{18}$$

for $j = 1, \ldots, J; t \in \{8, 15, 22, 29, 36\}$, where α_j and β_j denote random effects per rat. The model M_A random effects and per-rat effects are modeled hierarchically,

$$\alpha_j \mid \mu_\alpha, \sigma_\alpha \sim \mathcal{N}\left(\mu_\alpha, \sigma_\alpha^2\right), \beta_j \sim \mathcal{N}\left(\mu_\beta, \sigma_\beta^2\right) \tag{19}$$

with hyper-priors $\mu_\alpha \sim \mathcal{N}(250, 20)$, $\mu_\beta \sim \mathcal{N}(6, 2)$, $\sigma_\alpha \sim$ Gamma $(25, 2)$, and $\sigma_\beta \sim$ Gamma $(5, 10)$. The observation noise prior is $\sigma_y \sim \text{Gamma}(1, 2)$. Prior parameters were chosen using prior predictive checks.

In this example, we use parallel CV to check whether the random effect (i.e. rat-specific slope β_j) does a better job of predicting the weight of a new rat, than if a common β had been used. The CV scheme leaves a rat out for each fold, for a total of $K = 30$ folds. The alternative model is

$$M_B : y_{j,t}|\alpha_j, \beta, \sigma^2 \sim \mathcal{N}\left(\alpha_j + \beta t, \sigma^2\right), \tag{20}$$

for $j = 1, \ldots, J; t \in \{8, 15, 22, 29, 36\}$, where the prior $\beta \sim \mathcal{N}(6, 2)$ was chosen using prior predictive checks (Gelman et al. 2014).

Figure 6 shows that the PCV results have stabilized by 1000 iterations, and Pr $(M_A \succ M_B) \approx 90\%$. On an NVIDIA T4 GPU, PCV with 480 chains targeting all 60 posteriors took 18 s, which included a 10 s warm-up phase (Table C2 in Appendix C). Full-data inference took 11 and 15 s, respectively, which suggests naive brute force CV would take about 13 min. $\widehat{R}_{\max}$ plots suggest convergence after around 500 iterations per chain (Figure C2, Appendix C). In contrast, many chains still exceeded the 1.01 benchmark for $\widehat{\text{R}}$ even after 2, 000 iterations (Figure C1, Appendix C).

Example 3 (Home radon) Radon is a naturally-occurring radioactive element that is known to cause lung cancer in patients exposed to sufficiently high concentrations. Gelman and Hill (2006) present a hierarchical model of radon concentrations in U.S. homes. The data cover $N_D = 12{,}573$ homes in $J = 386$ counties. For our purposes we will assume that the goal of the model is to predict the level of radon in U.S. counties, including those not in the sample (i.e. out-of-sample county-wise prediction). The authors model the level of radon y_i in the ith house as normal, conditional on a random county effect α_j and the floor of the house x_i where the measurement was taken. We will compare two model formulations:

$$M_A : y_i \mid \alpha, \beta, \sigma_y^2 \sim \mathcal{N}\left(\alpha_{j[i]} + \beta x_i, \sigma_y^2\right), \tag{21}$$

$$M_B : y_i \mid \alpha, \beta, \sigma_y^2 \sim \mathcal{N}\left(\alpha_{j[i]}, \sigma_y^2\right), \tag{22}$$

for $i = 1, \ldots, N_D,$, where β is a fixed effect, $\alpha_{j[i]}$ is the random effect for the county corresponding to observation i, and σ^2 is a common observation variance. For both models the county effect is modeled hierarchically,

$$\alpha_j \mid \mu_\alpha, \sigma_\alpha^2 \sim \mathcal{N}\left(\mu_\alpha, \sigma_\alpha^2\right), \tag{23}$$

for $j = 1, \ldots, J$. The remaining priors are chosen to be weakly informative, $\mu_\alpha \sim \mathcal{N}(0, 4)$ and $\sigma_\alpha^2 \sim \text{Gamma}(6, 9)$. The other parameter priors are $\beta \sim \mathcal{N}(0, 1)$ and $\sigma_y^2 \sim \text{Gamma}(10, 10)$. A non-centered parameterization is used for MCMC inference and the model is fit by HMC. Prior parameters were chosen using prior predictive checks.

We will use county-wise PCV to determine whether the floor measure improves predictive performance. The estimate $\Pr(M_A \succ M_B) \approx 100\%$ stabilizes quickly (Fig. 7).

The parallel inference procedure takes a total of 92 s to draw 2000 parallel MCMC iterations plus 2000 warmup iterations across a total of 3088 chains targeting 772 posteriors. The 386 fold posteriors are sampled consecutively for each model. This compares with 45 and 35 s for the full-data models (see Table C3 in Appendix C for details). At 35 s per fold, a naive implementation of brute force CV across all would have taken 7.5 h to run.

Example 4 (Passenger arrivals) Australia is an island nation, for which almost all migration is by air travel. Models of passenger arrivals and departures are useful for estimating airport service requirements, the health of the tourist sector, and economic growth resulting from immigration.

We compare two simple models of monthly international air arrivals to all Australian airports in the period 1985–2019, using data provided by the Australian Bureau of Transport and Infrastructure Research Economic (BITRE). The data are seasonal and nonstationary (Figure C5, Appendix C), so we model month-on-month (M_A) and year-on-year (M_B) changes with a seasonal autoregression. The power spectrum of the month/month growth rates display seasonality at several frequencies, while annual figures do not, suggesting that annual seasonality is present (Figure C6, Appendix C).

It is therefore natural to model these series using seasonal autoregressions on the month/month or year/year growth rates. Let $y_t \in \mathbb{R}$ denote the growth rate, observed monthly. We model

$$y_t = \sum_{i=1}^{p} \rho_i y_{t-i} + \beta_0 + \sum_{j=1}^{q} \beta_j d_j + \sigma \varepsilon_t, \tag{24}$$

for $t = 1, \ldots, T$, $\varepsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$ .. The noise standard deviation prior is $\sigma \sim \mathcal{N}^+(0, 1)$. For AR effects we impose the prior $(2\rho_i - 1) \sim \text{Beta}(5, 5)$ and for the constant and seasonal effects $\beta_j \sim \mathcal{N}(0, 1)$.

Parallel CV results stabilize after a few hundred iterations per chain (Fig. 8). PCV took a total of 6.1 s to draw to draw 500 iterations per chain (Table C4, Appendix C). Naively running all 790 models in succession would have taken about 4.4 h.

6 Discussion

We have demonstrated a practical workflow for conducting fast, general brute force CV in parallel on modern computing accelerator hardware. We have also contributed methods for implementing and checking the resulting output.

Our proposed workflow is a natural extension of standard Bayesian inference workflows for MCMC-based inference (Gelman et al. 2020), extended to include initialization of parallel MCMC chains and joint convergence assessment for the overall CV objective.

The use of parallel hardware enables significantly faster CV procedures (in wall clock time), and on a practical level represents a sharp improvement in both flexibility and speed over existing CPU-based approaches. In contrast to approximate CV methods, our approach reflects a transition from computing environments that are predominantly compute-bound (storage and bandwidth are not practically constrained), to a new era with fewer constraints on computing power but where memory and bandwidth are more limited. Efficient use of parallel hardware can in some cases reduce the energy required and associated carbon release for compute-heavy tasks, such as simulation studies involving repeated applications of CV.

Our proposed diagnostic criteria provide the analyst with tools for assessing convergence across a large number of estimated posteriors, alleviating the need to examine each posterior individually. Further efficiencies could be gained by developing formal stopping rules for halting inference when chains have mixed and the desired accuracy has been attained, although stopping rules should be applied with care as they can also increase bias (e.g., Jones et al. 2006; Cowles et al. 1999). We leave this to future research.

Moreover, online algorithms' frugal memory requirements carries advantages for several classes of users. Top-end GPUs can run larger models (and/or more folds simultane-

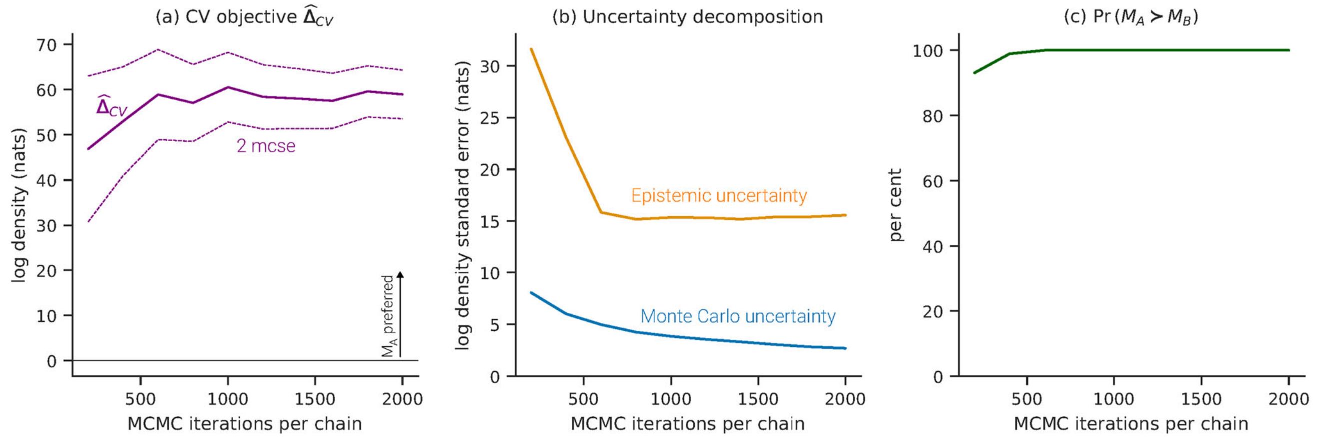

Fig. 7 Progressive estimates for groupwise cross-validation for Example 3 (home radon). Model M_A is preferred. The model selection statistic (**a**) is clearly positive and the probability that M_A predicts better than M_B stabilizes within a few hundred iterations

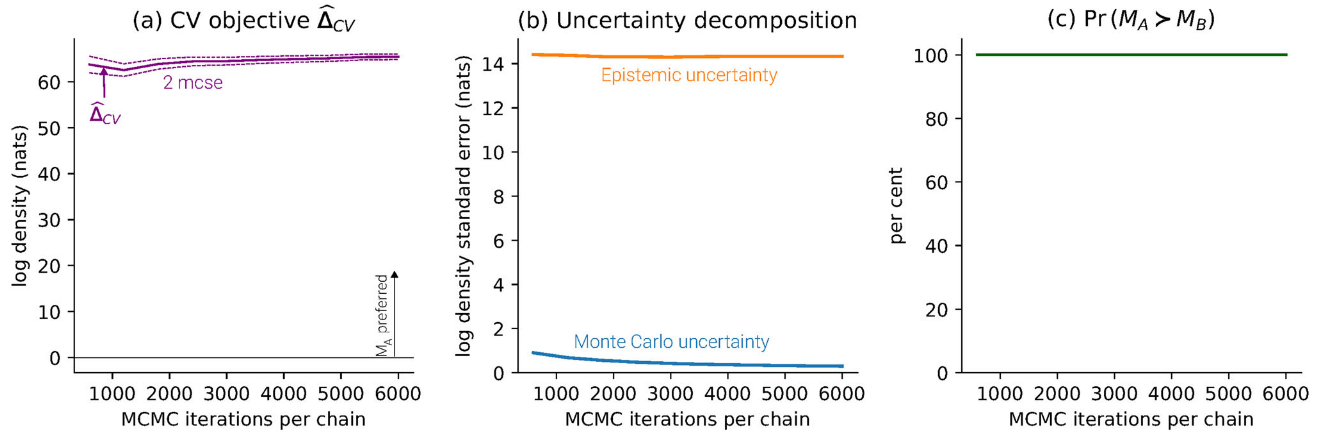

Fig. 8 Progressive estimates for groupwise cross-validation for Example 4 (passenger arrivals). The model selection statistic (a) is clearly positive and the probability of M_A predicting better than M_B stabilizes within a few hundred iterations

ously), while commodity computers (e.g. laptops with less capable integrated GPUs) can perform a larger range of useful tasks. Beyond accelerator hardware, our approach may have benefits on CPU-based architectures too, by exploiting within-core vector units and possibly improving processor cache performance because of the tight memory footprint of online samplers.

Two possible extensions to this work could further reduce memory footprint on accelerators. The adaptive subsampling approach of Magnusson et al. (2020) would require that only a subset of folds be estimated before a decision became clear. In addition, the use of stochastic HMC (Chen et al. 2014) would permit that only a subset of the dataset be loaded on the accelerator at any one time.

Further work could include adaptive methods to focus computational effort in the areas that would most benefit from further MCMC draws, for example on fold posteriors with the largest MC variance, as well as a stopping rule to halt inference when a decision is clear. Turn-key parallel brute-force CV routines would be a useful extension to probabilistic programming languages. Parallel CV can also be done using inference methods other than MCMC, such as variational inference.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11222-024-10404-w.

Acknowledgements The authors would like to thank Charles Margossian for helpful comments. AC's work was supported in part by an Australian Government Research Training Program Scholarship. AV acknowledges the Research Council of Finland Flagship program: Finnish Center for Artificial Intelligence, and Academy of Finland project (340721). CF acknowledges financial support under National Science Foundation Grant SES-1921523.

Author Contributions A.C. drafted the manuscript, developed experiments, and prepared the figures. A.C. and A.V. formulated the method. All authors edited and reviewed the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Declarations

Conflict of interest The authors declare no competing interests.

References

Chen, T., Fox, E., Guestrin, C.: Stochastic gradient Hamiltonian Monte Carlo. In: International conference on machine learning, PMLR, pp 1683–1691 (2014)

Cooper, A., Simpson, D., Kennedy, L., et al.: Cross-validatory model selection for Bayesian autoregressions with exogenous regressors Bayesian Anal. Advance Publication (2024). https://doi.org/10.1214/23-BA1409

Cowles, M.K., Roberts, G.O., Rosenthal, J.S.: Possible biases induced by MCMC convergence diagnostics. J. Stat. Comput. Simul. **64**(1), 87–104 (1999). https://doi.org/10.1080/00949659908811968

Dawid, A.P., Musio, M.: Theory and applications of proper scoring rules. METRON **72**(2), 169–183 (2014). https://doi.org/10.1007/s40300-014-0039-y

Dawid, A.P., Musio, M.: Bayesian model selection based on proper scoring rules. Bayesian Anal. **10**(2), 479–499 (2015). https://doi.org/10.1214/15-BA942

Dawid, A.P., Sebastiani, P.: Coherent dispersion criteria for optimal experimental design. Ann. Stat. **27**(1), 65–81 (1999)

Dillon, J.V., Langmore, I., Tran, D., et al.: TensorFlow distributions, (2017). arXiv:1711.10604

Gagnon, P., Maire, F., Zanella, G.: Improving multiple-try metropolis with local balancing. J. Mach. Learn. Res. **24**(248), 1–59 (2023)

Geisser, S.: The predictive sample reuse method with applications. J. Am. Stat. Assoc. **70**(350), 320–328 (1975). https://doi.org/10.1080/01621459.1975.10479865

Gelfand, A.E., Hills, S.E., Racine-Poon, A., et al.: Illustration of Bayesian inference in normal data models using Gibbs sampling. J. Am. Stat. Assoc. **85**(412), 972–985 (1990)

Gelfand, A.E., Dey, D.K., Chang, H.: Model determination using predictive distributions, with implementation via sampling-based methods (with discussion). Bayesian Stat. **4**, 147 (1992)

Gelman, A., Hill, J.: Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press (2006)

Gelman, A., Rubin, D.B.: Inference from iterative simulation using multiple sequences. Stat. Sci. **7**(4), 457–472 (1992)

Gelman, A., Carlin, J.B., Stern, H.S., et al.: Bayesian Data Analysis, 3rd edn. Chapman & Hall/CRC, Boca Raton (2014)

Gelman, A., Vehtari, A., Simpson, D., et al.: Bayesian workflow (2020). Preprint at arXiv:2011.01808,

Geyer, C.J.: Practical Markov chain Monte Carlo. Stat. Sci. pp 473–483 (1992)

Glatt-Holtz, N.E., Holbrook, A.J., Krometis, J.A., et al.: Parallel MCMC algorithms: Theoretical foundations, algorithm design, case studies (2022). arXiv preprint arXiv:2209.04750

Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. J. Am. Stat. Assoc. **102**(477), 359–378 (2007). https://doi.org/10.1198/016214506000001437

Hoffman, M.D., Gelman, A., et al.: The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. J. Mach. Learn. Res. **15**(1), 1593–1623 (2014)

Holbrook, A.J.: A quantum parallel Markov chain Monte Carlo. J. Comput. Graph. Stat. **32**, 1402 (2023)

Hyvärinen, A., Dayan, P.: Estimation of non-normalized statistical models by score matching. J. Mach. Learn. Res. **6**(4), 695 (2005)

Jones, G.L., Haran, M., Caffo, B.S., et al.: Fixed-width output analysis for Markov chain Monte Carlo. J. Am. Stat. Assoc. **101**(476), 1537–1547 (2006). https://doi.org/10.1198/016214506000000492

Krüger, F., Lerch, S., Thorarinsdottir, T., et al.: Predictive inference based on Markov chain Monte Carlo output. Int. Stat. Rev. **89**(2), 274–301 (2021). https://doi.org/10.1111/insr.12405

Kumar, R., Carroll, C., Hartikainen, A., et al.: ArviZ: a unified library for exploratory analysis of Bayesian models in python. J. Open Source Softw. (2019)

Lao, J., Louf, R.: Blackjax: A sampling library for JAX, (2020). http://github.com/blackjax-devs/blackjax

Lao, J., Suter, C., Langmore, I., et al.: tfp.mcmc: Modern Markov Chain Monte Carlo Tools Built for Modern Hardware (2020). Preprint at arXiv:2002.01184

Magnusson, M., Vehtari, A., Jonasson, J., et al.: Leave-one-out cross-validation for Bayesian model comparison in large data. In: Chiappa S, Calandra R (eds) Proceedings of the twenty third international conference on artificial intelligence and statistics, proceedings of machine learning research, vol 108. PMLR, pp 341–351 (2020), https://proceedings.mlr.press/v108/magnusson20a.html

Mahoney, M.J., Johnson, L.K., Silge, J., et al.: Assessing the performance of spatial cross-validation approaches for models of spatially structured data (2023). Preprint at arXiv:2303.07334

Margossian, C.C., Hoffman, M.D., Sountsov, P., et al.: Nested $\hat{R}$: Assessing the convergence of Markov chain Monte Carlo when running many short chains (2023). Preprint at arXiv:2110.13017

Moins, T., Arbel, J., Dutfoy, A., et al.: On the use of a local $\hat{r}$ to improve mcmc convergence diagnostic. Bayesian Anal. Advance Publication. (2023) https://doi.org/10.1214/23-BA1399

Neal, R.M.: MCMC using Hamiltonian dynamics. In: Handbook of Markov Chain Monte Carlo, vol. 2, pp. 113–162. CRC Press, New York (2011)

Neiswanger, W., Wang, C., Xing, E.P.: Asymptotically exact, embarrassingly parallel MCMC. In: Proceedings of the Thirtieth conference on uncertainty in artificial intelligence. AUAI Press, Arlington, Virginia, USA, UAI'14, p 623–632 (2014)

Nissen, J.N.: What scientists must know about hardware to write fast code. https://viralinstruction.com/posts/hardware/, Accessed 09 Jan 2023 (2023)

Racine, J.: Consistent cross-validatory model-selection for dependent data: hv-block cross-validation. J. Econ. **99**(1), 39–61 (2000). https://doi.org/10.1016/S0304-4076(00)00030-0

Robert, C.P., Casella, G.: Monte Carlo Statistical Methods. Springer, New York (2004)

Roberts, D.R., Bahn, V., Ciuti, S., et al.: Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography **40**(8), 913–929 (2017)

Roy, V.: Convergence diagnostics for Markov chain Monte Carlo. Ann. Rev. Stat. Appl. **7**, 387–412 (2020)

Rubin, D.B.: The Bayesian bootstrap. Ann. Stat. **9**(1), 130–134 (1981). https://doi.org/10.1214/aos/1176345338

Scott, S., Blocker, A., Bonassi, F., et al.: Bayes and big data: the consensus Monte Carlo algorithm. Int. J. Manag. Sci. Eng. Manag. **11**(2), 78–88 (2016). https://doi.org/10.1080/17509653.2016.1142191

Sivula, T., Magnusson, M., Matamoros, A.A., et al.: Uncertainty in Bayesian leave-one-out cross-validation based model comparison, (2022). Preprint at arXiv:2008.10296

Stallings, W.: Computer Organization and Architecture. Pearson Education, Limited (2015)

Stan Development Team: Stan Modeling Language: User's Guide and Reference Manual (2022)

Vehtari, A., Lampinen, J.: Bayesian model assessment and comparison using cross-validation predictive densities. Neural Comput. **14**(10), 2439–2468 (2002). https://doi.org/10.1162/08997660260293292

Vehtari, A., Gelman, A., Gabry, J.: Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Statist. Comput. **27**(5), 1413–1432 (2017). https://doi.org/10.1007/s11222-016-9696-4

Vehtari, A., Gelman, A., Simpson, D., et al.: Rank-normalization, folding, and localization: an improved $\widehat{R}$ for assessing convergence of MCMC. Bayesian Anal. **1**(1), 1–26 (2020). https://doi.org/10.1214/20-ba1221

Vehtari, A., Gelman, A., Sivula, T., et al.: Expectation propagation as a way of life: a framework for Bayesian inference on partitioned data. J. Mach. Learn. Res. **21**(17), 1–53 (2020)