Lambert, Ben; Vehtari, Aki

R* : A Robust MCMC Convergence Diagnostic with Uncertainty Using Decision Tree Classifiers

Published in:
Bayesian Analysis

DOI:
10.1214/20-BA1252

Published: 01/06/2022

Document Version
Publisher's PDF, also known as Version of record

Bayesian Analysis (2022) **17**, Number 2, pp. 353–379

R^*: A Robust MCMC Convergence Diagnostic with Uncertainty Using Decision Tree Classifiers

Ben Lambert[*] and Aki Vehtari[†]

Abstract. Markov chain Monte Carlo (MCMC) has transformed Bayesian model inference over the past three decades: mainly because of this, Bayesian inference is now a workhorse of applied scientists. Under general conditions, MCMC sampling converges asymptotically to the posterior distribution, but this provides no guarantees about its performance in finite time. The predominant method for monitoring convergence is to run multiple chains and monitor individual chains' characteristics and compare these to the population as a whole: if within-chain and between-chain summaries are comparable, then this is taken to indicate that the chains have converged to a common stationary distribution. Here, we introduce a new method for diagnosing convergence based on how well a machine learning classifier model can successfully discriminate the individual chains. We call this convergence measure R^*. In contrast to the predominant $\widehat{R}$, R^* is a single statistic across all parameters that indicates lack of mixing, although individual variables' importance for this metric can also be determined. Additionally, R^* is not based on any single characteristic of the sampling distribution; instead it uses all the information in the chain, including that given by the joint sampling distribution, which is currently largely overlooked by existing approaches. We recommend calculating R^* using two different machine learning classifiers — gradient-boosted regression trees and random forests — which each work well in models of different dimensions. Because each of these methods outputs a classification probability, as a byproduct, we obtain uncertainty in R^*. The method is straightforward to implement and could be a complementary additional check on MCMC convergence for applied analyses.

1 Introduction

Markov chain Monte Carlo (MCMC) is the class of exact-approximate methods that has contributed most to applied Bayesian inference in recent years. In particular, MCMC has made Bayesian inference widely available to a diverse community of practitioners through the many software packages that use it as an internal inference engine: from Gibbs sampling (Geman and Geman, 1984), which underpins the popular BUGS (Lunn et al., 2000) and JAGS (Plummer et al., 2003) libraries, to more recent algorithms: for example, Hamiltonian Monte Carlo (HMC) (Neal et al., 2011), the No U-Turn Sampler (NUTS) (Hoffman and Gelman, 2014), and a dynamic HMC variant (Betancourt, 2017), which Stan (Carpenter et al., 2017), PyMC3 (Salvatier et al., 2016), Turing (Ge et al., 2018), TensorFlow Probability (Dillon et al., 2017) and Pyro (Bingham et al.,

arXiv: 2003.07900

[*]MRC Centre for Global Infectious Disease Analysis, School of Public Health, Imperial College London, W2 1PG, United Kingdom, ben.c.lambert@gmail.com

[†]Department of Computer Science, Aalto University, Finland, aki.vehtari@aalto.fi

https://doi.org/10.1214/20-BA1252

2019) implement. MCMC methods are currently the most effective tools for sampling from many classes of posterior distributions encountered in applied work, and it seems unlikely that this trend will change soon.

Its importance in applied scientists' toolkits means it is essential that MCMC is used properly and with adequate care. A cost of automated inference software is that it is increasingly easy to regard MCMC as oracular: giving uncompromised views onto the posterior. Because of this, software packages (Stan (Carpenter et al., 2017), for example), go to great lengths to communicate to users any issues with sampling.

The most important determination of whether MCMC has worked is how closely the sampling distribution has converged to the posterior (Brooks et al., 2011). MCMC methods are thus created because of an asymptotic property: that given an infinite number of draws, their sampling distribution approaches the posterior (under general conditions). Although the guarantees are asymptotic, MCMC estimates can have negligible bias with only a relatively small number of draws.

The one diagnostic method for determining whether practical convergence has occurred relies on the fact that the posterior distribution is the unique stationary distribution for an MCMC sampler. Therefore, it would appear that, if an MCMC sampling distribution stops changing, then convergence has occurred. Unfortunately, anyone who uses MCMC knows that it is full of false dawns: chains can easily become stuck in areas of parameter space, and observation over short intervals mean the sampling distribution *appears* converged (Gelman and Rubin, 1992b). Like furious bees trapped in a room of a house (Lambert, 2018b), MCMC samplers may fail to move due to the narrow gaps that join neighbouring areas. With MCMC, absence of evidence of new areas of high posterior density is, time and again, not evidence of their absence.

To combat this curse of hindsight, running multiple, independent chains, which have been initialised at diverse areas of parameter space is recommended (Gelman and Rubin, 1992a). If the chains appear not to "mix" – a term essentially meaning that it is difficult to resolve an individual chain's path from the mass of paths overlaid on top of one another – they are yet to converge. This approach makes it less likely that faux-convergence will occur due to chains becoming stuck in an area of parameter space, and running multiple chains is standard practice in applied inference (Lambert, 2018a). The predominant approach to quantitatively measuring this mixing is to compare each chain's sampling distribution to that of the population of chains as a whole: specifically, $\widehat{R}$ – the main convergence statistic used – compares within-chain variance to that between-chains (Gelman and Rubin, 1992a). If these variances are similar, $\widehat{R} \approx 1$, and chains are deemed to have mixed. Recently, Stan has adopted more advanced variations on the original $\widehat{R}$ formula: for example, splitting individual chains in two to combat poor intra-chain mixing (Gelman et al., 2013); and using ranks of parameter draws rather than the raw values themselves to calculate $\widehat{R}$ (Vehtari et al., 2020). Additionally, there has been more focus on ensuring that the effective sample size (ESS), a measure of sample quality (see, for example, Lambert (2018a)), is sufficient, and accordingly, new measures of this quantity have been proposed (Vehtari et al., 2020) and adopted (Carpenter et al., 2017). Collectively, these statistics help alert users of MCMC to issues with sampling (that typically echo issues with the model) meaning that all is not hunky dory.

Here, we introduce R^*, a new convergence diagnostic metric. This statistic is built on the intuition that, if chains are mixed, it should not be possible to discern from a draw's value the chain that generated it. Rephrased, it should not be possible to predict which draws come from which chain. In this vein, we use machine learning (ML) classifiers to measure convergence. Specifically, we train classifiers to predict the chain that generated each observation. By evaluating the performance of classifiers on a held-out test set, this provides a new convergence metric. To maximise predictive accuracy, our chosen classifiers naturally exploit differences in the full joint distributions between chains, which means they are sensitive to variations across the joint distribution of target model dimensions unlike most existent convergence diagnostics. Our statistic, unlike its $\widehat{R}$ cousins, is scalar-valued for multivariate distributions: one model provides a single R^*, whereas $\widehat{R}$ has separate values for each univariate marginal distribution. However, the ML classifiers we use can straightforwardly be interrogated to estimate which parameters were most important for generating predictive accuracy.

There are, of course, a huge variety of possible ML classifiers. For a method to be useful and widely adopted, however, it needs to satisfy a number of criteria: first, across a range of examples, it should consistently be able to detect poor MCMC convergence; second, the ML classifier should be trainable in a reasonable amount of time; lastly, the methods should be tuned (via their hyperparameters) so as to be standardised, so that R^* computed by one analyst is comparable to that computed by another. Here, we perform comparisons across some of the most popular methods in the ML literature and recommend two tree-based ML classifiers: gradient-boosted regression trees (Friedman, 2001; Greenwell et al., 2019) ("GBM") and random forest classifiers (Breiman, 2001) ("RF"). Both of these methods performed consistently well across our range of examples, and the models were relatively efficient to train. We recommend calculating R^* using both of these methods: GBMs tend to perform best for low dimensional cases; in moderate dimensional cases upwards, RF dominates. This difference in performance is partly due to hyperparameter tuning: echoing previous results in the literature (Boehmke and Greenwell, 2019, Chapters 11&12), our results show that GBMs, in general, are more sensitive to hyperparameter choice than are RFs. For a GBM classifier to perform well in higher dimensions, more rounds of boosting are required, which substantially increases training time; instead, we fix the hyperparameters of GBMs to specific values to ensure practical usefulness. For RF classifiers, empirical studies have demonstrated reliable heuristics for selecting robust hyperparameters (Bernard et al., 2009), and, in any case, across our set of test cases, these classifiers are relatively insensitive to hyperparameter choice.

For the types of problem we tested, R^* calculation is of a speed comparable to some of the newer $\widehat{R}$ measures calculated (typically $\mathcal{O}$ (seconds) to calculate), although for models with 10,000s of parameters and many iterations, the time taken is longer. In addition, since ML classifiers can output predicted class probabilities, we obtain uncertainty measures for R^*, which we find provides a useful summary of MCMC convergence. R^* can straightforwardly be incorporated into existing software libraries to provide a complementary convergence metric alongside more established measures.

The structure of this paper is as follows: in §2, we describe in detail the method for calculating R^* and its uncertainty; in §3, we examine the performance of R^* across a range of scenarios. Code for reproducing the analyses is provided at `https://github.com/ben18785/ml-mcmc-convergence`.

2 Method

If Markov chains have not mixed, it is possible to guess (with more accuracy than chance) to which chain a draw belongs from its value. This is possible if there are differences in the sampling distribution for any dimension in the target distribution (Figure 1): in this case, if the marginal distributions differ between chains, this information can be used to predict which chain a draw belongs to. It is also possible to predict the chain that generated a given draw if there are differences in the joint distribution of two (or more) dimensions of the target, even if the marginal distributions are the same (Figure 2).

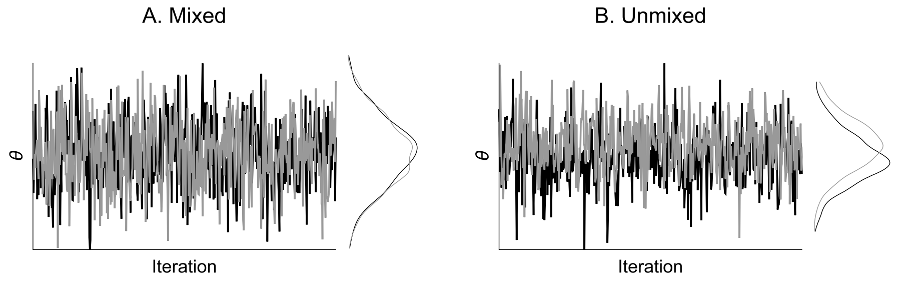

Figure 1: **Chain prediction based on the marginal distribution of a single parameter.** A shows the path of two chains that have mixed (with marginal distribution to the right of panel); B shows two chains that have not mixed.

These two cases, whilst simple, illustrate the basis of our approach. To determine if a set of Markov chains has converged to the same distribution, we train a supervised ML model to classify the chain to which each draw belongs. By evaluating its performance on a separate test set, we delineate whether chains have mixed based on whether classification accuracy is above the "null" case, where accuracy is $1/N$, and N is the number of chains. By taking the ratio of classification accuracy to this null accuracy, we obtain a statistic that is interpretable in a similar way to $\widehat{R}$ (Vehtari et al., 2020). In a nod to this established statistic, we call our statistic R^*, and, by design, $R^* \approx 1$ signifies convergence. Algorithm 1 gives a recipe for calculating R^*.

To identify promising candidate ML methods, we run a series of experiments using a number of popular classifiers (see §S7.1). Two methods performed consistently well across our examples: gradient-boosted regression trees (also known as a type of gradient-boosted machine or GBM, introduced in Friedman, 2001) and random forest classifiers (Breiman, 2001) ("RF"). Both of these approaches are based on decision trees: GBMs use an iterative approach known as "boosting", where each subsequent decision tree aims

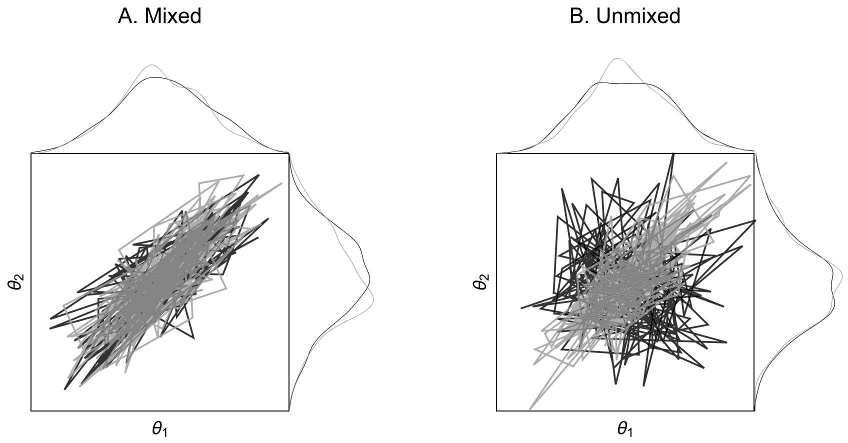

Figure 2: **Chain prediction based on the joint distribution of two parameters where each chain's marginals are the same.** A shows the path of two chains that have mixed resulting in similar sampling distributions (to the right and above each panel); B shows two chains that have not mixed.

to predict the residuals from the previous one; RFs use an approach known as "bagging" which trains decision trees on many bootstrapped copies of the training data, yielding a collection of independently trained trees whose individual decisions, when aggregated, yield a class. For the implementations of these classifiers, we use those available in **R**'s "Caret" package (Kuhn et al., 2008), which, in turn, uses the "gbm" (Greenwell et al., 2019) and "randomForest" (Liaw and Wiener, 2002) packages.

The data for each chain have dimensions: $X \in \mathbb{R}^S \times \mathbb{R}^K$, where S is the number of draws taken (here assumed the same for each chain, but this is not a binding constraint), and K is the number of parameters. We split each chain's draws into randomly divided training and testing sets: here, we use 70% of draws for training and 30% for testing. Both approaches typically took $\mathcal{O}$ (seconds) on a desktop computer to execute training then prediction on a testing set for most models we consider in §3.

Since different posteriors present different challenges to MCMC samplers, the nature of classification boundaries is problem-specific. Because of this, there is no unique optimal classifier across all problems, and the performance of the algorithms we investigate depends on their hyperparameters (see §S7.2). GBM models have a number of hyperparameters and have previously been demonstrated to be hard to tune (Boehmke and Greenwell, 2019, Chapter 12). To ensure practical run time for this method, we suggest running it using a default hyperparameter set: an interaction depth of 3, a shrinkage parameter of 0.1, 10 observations being the minimum required for each node, and that 50 trees be grown. This choice of hyperparameters was chosen to present a balance between training cost and classification accuracy, producing an R^* that was a stringent measure of convergence — in general, more stringent than $\widehat{R}$. RFs are generally less sensitive to variation in their single hyperparameter: m_{try}, the size of the subset of all features

Algorithm 1 R^* calculation

```
Given chain-wise draws from the target, $\{X^{\{1\}}, X^{\{2\}}, \ldots, X^{\{N\}}\}$ and a test set length, $S_{\text{test}}$:
for $m = 1$ to $N$ do
    Create train and test sets by random-sampling (w/o replacement), $X^{\{m\}} \rightarrow \{X^{\{m\}}_{\text{train}}, X^{\{m\}}_{\text{test}}\}$
end for
Stack $X_{\text{train}} = (X^{\{1\}}_{\text{train}}, X^{\{2\}}_{\text{train}}, \ldots, X^{\{N\}}_{\text{train}})^T$
Stack $X_{\text{test}} = (X^{\{1\}}_{\text{test}}, X^{\{2\}}_{\text{test}}, \ldots, X^{\{N\}}_{\text{test}})^T$
Train ML model to classify chain id from any draw, $x$: $\text{ML}(x|X_{\text{train}}) \rightarrow c$
for $s = 1$ to $S_{\text{test}}$ do
    Obtain test draw, $x^{\{s\}} = X_{\text{test}}(s) \in \mathbb{R}^K$
    Predict chain id, $c^{\{s\}} = \text{ML}(x^{\{s\}}|X_{\text{train}})$
    Compare with actual id, $c^s$: $a^{\{s\}} = \mathbb{1}(c^{\{s\}} = c^s)$
end for
Calculate predictive accuracy, $\bar{a} = \frac{1}{S_{\text{test}}} \sum_{s=1}^{S_{\text{test}}} a^{\{s\}}$
Calculate ratio to null model accuracy, $R^* = \bar{a}/(1/N) = N\bar{a}$
return $R^*$
```

over which to search for an optimal split (Boehmke and Greenwell, 2019, Chapter 11). Our experiments replicate these results (see §S7.2), and we suggest running RFs using the heuristic $m_{\text{try}} = \sqrt{K}$, which has previously been shown to be a choice resulting in robust classification performance (Bernard et al., 2009; Boehmke and Greenwell, 2019). Unless otherwise stated, in the examples explored in §3, the hyperparameters for these two classifiers were set at these defaults.

From a classifier fit, predicted chain probabilities can also be obtained, which we leverage to produce an uncertainty distribution for R^*. Algorithm 2 gives a recipe for generating draws from this distribution, which we now elaborate on in words. For each draw, s, in our testing set, classifiers output a simplex of chain probabilities: $\boldsymbol{p}^{\{s\}} = (p_1^{\{s\}}, p_2^{\{s\}}, \ldots, p_N^{\{s\}})$, which forms a categorical distribution that can be sampled from to yield a unique chain prediction, $c^{\{s\}}$. By comparing this classification to the true classification, c^s, we obtain a binary measure, $a^{\{s\}} = \mathbb{1}(c^{\{s\}} = c^s)$, of whether this prediction was correct. We repeat this process for each draw in the testing set, generating $\boldsymbol{a} = (a^{\{1\}}, a^{\{2\}}, \ldots, a^{\{S_{\text{test}}\}})$, whose average yields a single $R^{*\{i\}} = N\bar{a}$ estimate for iteration i. We then iterate this process, for $i = 1, 2, \ldots, I$, producing a set of $(R^{*\{1\}}, R^{*\{2\}}, \ldots, R^{*\{I\}})$, which collectively represent a distribution for R^*.

3 Results

To illustrate the versatility of R^*, we use a range of examples that demonstrate how this statistic fares across a range of scenarios. Table 1 summarises the examples and provides a rationale for their inclusion. The experiments not detailed in the main text are briefly described in §3.5 and more fully in the relevant sections given in Table 1.

Algorithm 2 Procedure to generate I draws of R^*

```
Given test data $X_{\text{test}}$, number of chains $N$, number of iterations $I$, and fitted
model, $\text{ML}(x|X_{\text{train}}) \to (p_1, p_2, \ldots, p_N)$:
for $i = 1$ to $I$ do
  for $s = 1$ to $S_{\text{test}}$ do
    Obtain test draw, $x^{\{s\}} = X_{\text{test}}(s) \in \mathbb{R}^K$
    Predict chain id probabilities, $(p_1^{\{s\}}, p_2^{\{s\}}, \ldots, p_N^{\{s\}}) = \text{ML}(x^{\{s\}}|X_{\text{train}})$
    Draw a chain id, $c^{\{s\}} \sim \text{categorical}(p_1^{\{s\}}, p_2^{\{s\}}, \ldots, p_N^{\{s\}})$
    Compare with actual id, $c^s$: $a^{\{s\}} = \mathbb{1}(c^{\{s\}} = c^s)$
  end for
  Calculate predictive accuracy, $\bar{a} = \frac{1}{S_{\text{test}}} \sum_{s=1}^{S_{\text{test}}} a^{\{s\}}$
  Calculate ratio to null model accuracy, $R^{*\{i\}} = \bar{a}/(1/N) = N\bar{a}$
end for
return $(R^{*\{1\}}, R^{*\{2\}}, \ldots, R^{*\{I\}})$
```

In all cases where $\widehat{R}$ was calculated, unless otherwise stated, we followed the approach in Vehtari et al. (2020) by calculating it as the maximum of rank-normalised split-$\widehat{R}$ and rank-normalised folded split-$\widehat{R}$: for simplicity, we refer to this as rank-normalised split-$\widehat{R}$.

3.1 Heterogeneity in chain variance: autoregressive example

In this section, we use a simple example to illustrate how R^* works. Specifically, we show how R^* can detect heterogeneous variance across Markov chains. This example aims to illustrate the basic mechanics behind how R^* works, so we use only the GBM classifier here. This section also investigates the sensitivity of this measure: across different training and testing sets (§3.1.2) and different draws from the ML model-predicted probability simplex (§3.1.3); and to differing numbers of chains (§3.1.4). The experiments we use to study these issues are all of similar form to the following data generating process: four Markov chains are generated, where each samples from an autoregressive order 1 (AR(1)) process of the form,

$$X_t = \rho X_{t-1} + \epsilon_t, \tag{3.1}$$

where $\epsilon_t \overset{i.i.d.}{\sim} \text{normal}(0, \sigma)$, $\rho = 0.3$ and $t = 1, 2, \ldots, 2000$. Three of the chains share the same $\sigma = 1$, whereas the other chain has $\sigma = 1/3$, so that it has 1/3 of the (unconditional) standard deviation of the others.

3.1.1 Performance of R^*

To illustrate the consistency of R^*, we performed 1000 replicates where, in each case, we generated four $\{X_t\}$ series as described (i.e. where one chain has a lower variance). We then fit a GBM to a labelled training set. The fitted model is then used to classify

Example	**Relevance**	**Section**
Autoregressive	**Examining R^* and sensitivities to its calculation**	3.1
	Detecting heterogeneous chain variance using R^*	3.1.1
	Stochasticity in R^*	3.1.2
	Generating R^* uncertainty measure	3.1.3
	Sensitivity of R^* to number of chains	3.1.4
Multivariate normals	**Detecting convergence in joint distributions**	3.2
	Unconverged joint distribution in bivariate normal	3.2.1
	High correlations between dims in 250D normal	3.2.2
	Measuring contributions of variables to poor convergence	3.2.3
Cauchy	**Detecting convergence for long-tailed distributions**	3.3
	Comparing R^* and existing measures to objective convergence	3.3.1
Eight schools model	**Hierarchical Bayesian model slow convergence**	3.4
Wide multivariate normal	**Detecting convergence when # draws $\sim$ # dims**	S3
Non-stationary marginals	**Detecting time-varying sampling distributions**	S4
	Trends in mean across all chains	S4.1
	Trends in mean in a single dimension	S4.2
	Trends in covariance	S4.3
	Sensitivity of R^* to chain persistence	S4.4
Ovarian and prostate models	**Bayesian models with many parameters and multimodal posteriors**	S5
Discrete Markov model	**Evaluating R^* on discrete parameter models**	S6
	Small state-space	S6.1
	Large state-space	S6.2
Various	**Sensitivity of R^* to ML model**	S7
	Comparing different ML classifiers	S7.1
	Sensitivity of R^* to GBM and RF hyperparameters	S7.2
Multivariate normal & Student-t dists.	**Comparing GBM and RF classifiers**	S8
	Detecting convergence in joint distributions	S8.1
	Tail convergence	S8.2

Table 1: Summarising the example problems and reasons for their inclusion.

draws in an independent test set according to the chain which generated them. For each replicate, we then calculated R^* as described in Algorithm 1.

In Figure 3A, we show how a GBM fitted to one such replicate dataset classifies observations according to a draw's value. Unsurprisingly, since the fourth chain has a smaller variance, observations close to zero are likely to be classified as being generated by this chain.

In Figure 3B, we show that $R^* > 1$ for all replicates, indicating that the chains had not converged in all cases. In Figure 3C, we show rank-normalised split-$\widehat{R}$ for each replicate; as for R^*, this metric indicates the chains had not converged in all replicates because $\widehat{R} > 1.01$.

3.1.2 Stochasticity in R^*

Unlike $\widehat{R}$, R^* is a stochastic convergence measure due to randomness in creating training and testing sets (essentially a form of sampling variation) and randomness in the methods used to train the ML model. This means that even if the same sample is used, R^* will return a different value each time it is calculated if the pseudorandom seed is

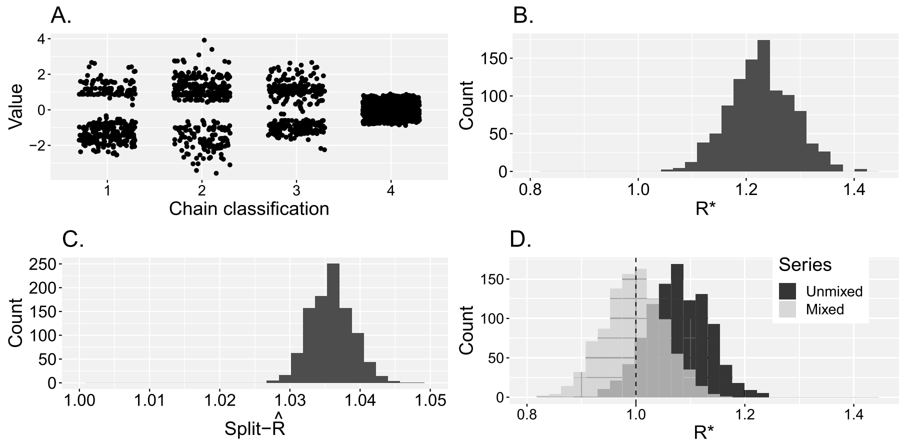

Figure 3: **Autoregressive example.** A shows how the GBM's classifications vary according to the draw's value for an example model fit; B shows R^* values generated by Algorithm 1 across 1000 replicate datasets; C shows corresponding rank-normalised split-$\widehat{R}$ values for each of the 1000 replicates; and D shows 1000 R^* samples as generated by Algorithm 2 for two series: the "unmixed" dataset being the same as used for figures A-C; the "mixed" dataset where all chains have the same distribution as described in §3.1. Note that, in D, only a single series of each series type is used to generate distribution. All examples used a GBM for classification using the default hyperparameter values given in §S2.

not fixed. To probe the extent of this randomness, we generated data using the same process as in §3.1 but now using varying sample sizes, including samples consisting of 500, 1000, 2000, 4000 and 8000 draws. For each dataset, we computed R^* on it 100 times, allowing the pseudorandom seed to vary between calculations. We stress that, for each sample size, we used the same dataset (so there were 5 datasets created in total – one for each sample size), so stochasticity comes from R^* calculation, not that from the data generating process.

In Figure 4, we show the results of this study. In this figure, the horizontal axis shows the sample size, and the vertical axis, the value of R^* in each repetition. This shows that as the number of samples increased, variation in R^* declined. At a sample size of 500, there were four cases where $R^* < 1$; in larger samples, there were none. Intuitively, the reduction in sampling variation when composing training and test sets from larger samples results in lower variability in ML model predictions. We also expect that larger sample sizes should lead to higher R^* values with lower variance, since more training data leads to ML models with lower generalisation error. We may start to see this here, since the median $R^* = 1.25$ at a sample size of 8000 was greater than for the smaller samples.

If randomness in R^* calculation leads to different conclusions about convergence being drawn, this would be problematic. One potential remedy for this is to repeatedly

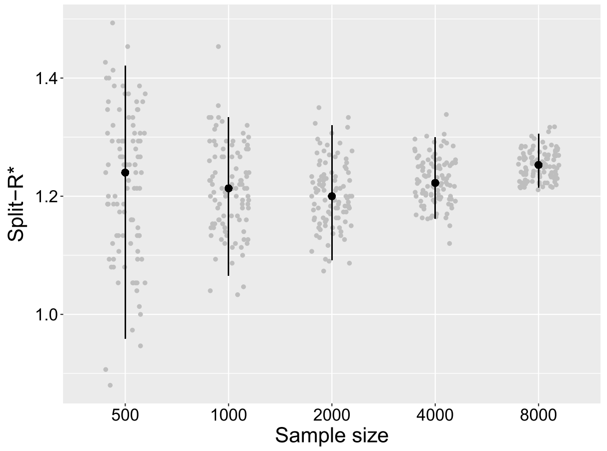

Figure 4: **Autoregressive example: R^* stochasticity.** The horizontal axis shows sample size; the vertical axis shows the value of R^* calculated as per Algorithm 1 applied to chains split into two halves. Grey points show the value of R^* for each replicate (jitter was added to point positions). Black points show the median R^* value; upper and lower whiskers show 2.5% and 97.5% quantiles. For each sample size, a single dataset was created and used for all R^* calculations.

calculate R^* on a given sample, much as we have done here, and consider the distribution of R^* values computed. The computational cost of doing this may, of course, be unreasonable. Instead, in §3.1.3, we consider an alternative approach based on bootstrapping a single ML model's predictions.

3.1.3 Uncertainty distribution for R^*

GBMs return a probability simplex for each draw indicating the probability that the draw was generated by a given chain. We can use this simplex to generate a measure of uncertainty in R^* as detailed in Algorithm 2. We demonstrate this idea using two datasets: one generated as described in §3.1, where one chain (out of four) has a lower variance than the others (we call this the "unmixed" data); and another, where all chains sample from the same distribution (we call this the "mixed" data). In Figure 3D, we show the R^* distributions in each case. For the unmixed data, the distribution has its bulk of mass away from 1 indicating lack of convergence. For the mixed data, the distribution is centred on 1 indicating convergence. In the mixed case, there are many draws where $R^* < 1$: these indicate that, in that particular draw from the probability simplex, chain classification is actually worse than selecting a chain identification uniformly at random. Much like how it is possible for $\hat{R} < 1$, this is a sample property, driven by the sampling distribution of the categorical distribution defined by the probability simplex.

It is worth emphasising that the uncertainty distribution obtained by Algorithm 2 differs from that obtained from repeatedly calculating R^* via Algorithm 1 as was done in §3.1.2. In Algorithm 2, variation in R^* comes from sampling from the probability simplex: if predicted chain probabilities are close to uniform, there will be greater uncertainty in R^*. Repeatedly calculating R^* by applying Algorithm 1 to the same dataset yields a distribution whose width derives from sampling variation when forming training and testing sets and the stochasticity in training ML models. Collectively, these differences mean that the two measures of uncertainty will differ.

There is an additional difference, though, in the central points of each distribution: the distribution obtained by Algorithm 2 will, in general, have a lower mean than that obtained by repeated application of Algorithm 1. To see this, note that the darker-shaded R^* distribution in Figure 3D was generated via Algorithm 2 and has a mean around 1.07; the distribution shown in Figure 3B was generated by repeatedly recomputing R^* using Algorithm 1 and has a mean closer to 1.22. This difference in mean is expected since predictive performance when assigning chain identities stochastically when sampling from the categorical distribution of the probability simplex (as is done in Algorithm 2) will generally result in worse prediction than when assigning each chain identity using whichever chain has the highest class probability (as is done in Algorithm 1). Of course, we would prefer it if the uncertainty distribution generated by Algorithm 2 had a mean closer to the one obtained by repeated application of Algorithm 1. Nonetheless, in practice, we have found that the mean of the R^* distribution generated by Algorithm 2 provides a useful cheaper diagnostic.

3.1.4 Sensitivity to number of chains

We have so far focused on the sensitivity of R^* to chain heterogeneity with a fixed number of chains: four. Since classification may become a harder problem when there are more categories, we now demonstrate how R^* (as calculated by Algorithm 1) performs across various numbers of chains. For comparison, we also illustrate how the performance of rank-normalised split-$\widehat{R}$ varies with number of chains. To do so, we consider an autoregressive example similar to that described in §3.1: where all chains bar one have $\sigma = 1$, and the remaining chain has $\sigma = 1/3$. We consider cases with 2, 4, 8, 16, and 32 chains. The other hyperparameters of the data generating process remain the same as in §3.1.

In Figure 5, we show the results of these simulations with 50 replicates at each number of chains. On the horizontal axis, we show the number of chains, and on the vertical axis, the value of each of the two convergence measures (R^* in the left panel; $\widehat{R}$ in the right panel). In general, both measures decline with chain count. For R^*, this may be because it is harder to classify chains when there are more of them. For $\widehat{R}$, this is because between-chain variance becomes relatively lower to that within them when there are more chains, and only one of them differs in its marginal distribution. Across the replicates we ran, median $\widehat{R} < 1.01$ for 16 or more chains; a minimum of $R^* = 1.10$ was obtained for 32 chains.

The decline of both of these measures when more chains are used hints that perhaps a moving threshold for diagnosing convergence may be pertinent to avoid neglecting those minority of chains with differing information. Here, however, we do not make suggestions on what such guidelines could be and leave this for later work.

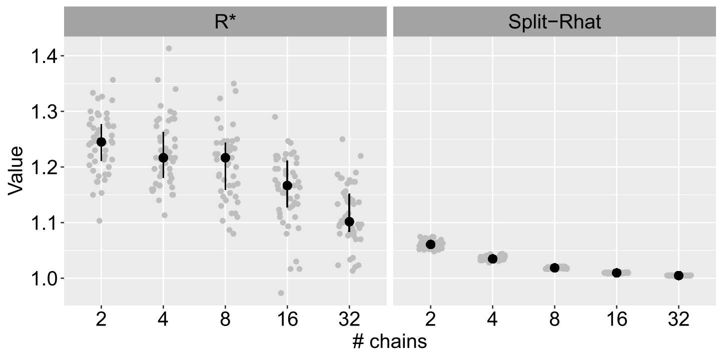

Figure 5: **Autoregressive example: sensitivity to number of chains.** The horizontal axis shows the number of chains used in the data generating process described in §3.1.4. The vertical axis shows the value of R^* as calculated by Algorithm 1 (left panel) on chains split into two halves, and rank-normalised split-$\widehat{R}$ (right panel). Grey points indicate the values of both convergence measures calculated for each replicate; horizontal jitter has been added to points. The point-ranges shown indicate the 25%, 50% and 75% quantiles across 50 replicates at each number of chains.

3.2 Diagnosing convergence in joint distributions: multivariate normal models

In this section, we illustrate how R^* can diagnose convergence issues in the joint target distribution.

3.2.1 Bivariate model

First, we consider a bivariate normal density. In all four chains, we use independent sampling to generate 2000 draws from bivariate normal densities with means of zero; in three of these chains, the covariance matrix is an identity matrix; in one chain, the covariance matrix also has unit diagonal terms but has off-diagonal terms of 0.9, indicating strong covariance between the two dimensions. By construction, all chains target the same marginal distribution in each dimension, but the fourth chain has a different joint distribution.

First, we use the code provided in Vehtari et al. (2020) to calculate rank-normalised $\widehat{R}$ and two different ESS measures that aim to capture how well certain regions of the posterior have been explored: these are known as bulk-ESS and tail-ESS. In all cases, the various quantities were calculated based on chains split into halves. For both dimensions, the two ESS measures were above 7000, and $\widehat{R} < 1.001$, indicating no issues with convergence.

Next, we estimate the R^* distribution using Algorithm 2 using both GBM and RF classifiers. These distributions are shown in Figure 6. The mean of the GBM-R^* distribution is 1.14, and >99% of R^* draws are above 1; the mean of the RF-R^* distribution

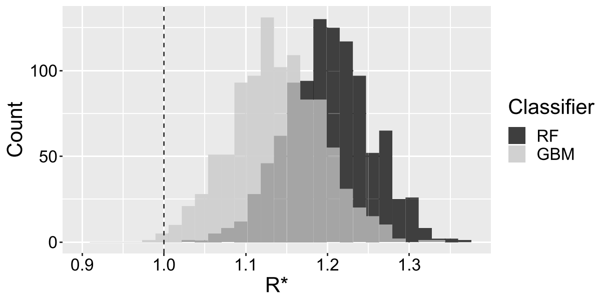

Figure 6: **Bivariate normal example.** The distribution for R^* across 1000 draws as calculated using Algorithm 2 for both the GBM and RF classifiers.

was 1.27 and all draws were above 1. Collectively, these measures indicate that the sampling distribution has not converged. By taking account of all the information in the chains, R^* is able to probe issues in joint distribution convergence which are missed by measures that consider only marginals.

3.2.2 250-dimensional model

We next consider a more challenging problem – a 250-dimensional multivariate normal target where its precision matrix, $\boldsymbol{A} \in \mathbb{R}^{250} \times \mathbb{R}^{250}$, is generated from a Wishart distribution (Hoffman and Gelman, 2014). We assume that the Wishart distribution's degrees of freedom is 250, resulting in a distribution with high correlations between dimensions. We use Stan's NUTS algorithm (Betancourt, 2017) to sample from this target distribution and run the algorithm for two different iteration counts (each time across 4 chains): 400 and 10,000 (the latter thinned by a factor of 5). First, we used Stan to sample from the "centered" parameterisation of this model, which is of the form,

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{A}^{-1}), \tag{3.2}$$

where $\boldsymbol{x} \in \mathbb{R}^{250}$. For each set of draws, we used Algorithm 2 with a GBM classifier to generate an uncertainty distribution for R^*, which is shown in Figure 7A (the equivalent plot for a RF classifier is similar and shown in Fig. S1). From the plot for the 400 iteration case, it is clear that convergence has not yet occurred since $R^* > 1$ across the bulk of this distribution. Even in the 10,000 iteration case, the R^* distribution remains stubbornly shifted a little rightwards of $R^* = 1$ (its mean is 1.06): in this case, $\widehat{R} < 1.01$ for all parameters (Figure 7B), although 54% had bulk-ESS < 400 and 13% of parameters had tail-ESS < 400 indicating issues with convergence (Vehtari et al., 2020).

Rather than run the MCMC sampler for more iterations, we move to a "non-centered" parameterisation, which introduces auxillary variables $\boldsymbol{z} \in \mathbb{R}^{250}$ that don't affect $p(\boldsymbol{x})$ but facilitate sampling from it. This model has the form,

$$\boldsymbol{A}^{-1} = \boldsymbol{L}\boldsymbol{L}^T, \qquad \boldsymbol{x} = \boldsymbol{L}\boldsymbol{z}, \qquad z_j \sim \text{normal}(0, 1), \text{ for } j = 1, 2, \ldots, 250. \tag{3.3}$$

where $\boldsymbol{L}$ is the Cholesky decomposition of the covariance matrix, $\boldsymbol{A}^{-1}$. Figure 7A shows the R^* distribution resultant from 10,000 NUTS iterations in this case: now the distribution has mean $R^* = 1.00$. Figure 7B shows the $\widehat{R}$ values for each x parameter in this model, and, echoing the result for R^*, $\widehat{R} < 1.01$ in all cases; further, bulk- and tail-ESS > 400 for all parameters.

3.2.3 Variable importance

In GBMs, it is possible to calculate variable importance (see, for example, Friedman, 2001 and Greenwell et al., 2019), which allows us to determine which variables were most informative for predictions. We now compare these with the more established metrics $\widehat{R}$ and ESS. For a GBM fitted to the centered model of eq. (3.2) with 10,000 MCMC iterations (thinning by a factor of 5) for each chain, we plot in Figure 7C variable importance (here high values mean a variable is more important) versus $\widehat{R}$ for all dimensions of the target distribution (including Stan's lp quantity, shown as a triangle). In this plot, there is a positive association between GBM's variable importance and $\widehat{R}$ (Spearman's rank correlation: $\rho = 0.17, S = 2185680, p < 0.01$). In Figure 7D, we plot variable importance versus two measures: bulk-ESS and tail-ESS, which both exhibited a strong non-linear negative association (Spearman's rank correlation: bulk-ESS: $\rho = -0.57, S = 4142470, p < 0.01$; tail-ESS: $\rho = -0.56, S = 4113709, p < 0.01$). Since none of these plots form perfect "lines" along which all the plotted points fall, this illustrates that variable importance provides information complementary to $\widehat{R}$ and ESS.

3.3 Infinite variance: Cauchy example

We next explore how R^* can be used to determine convergence for distributions with infinite variance. Like Vehtari et al. (2020), we first use Stan to sample from independent standard Cauchy distributions for each element of a 50-dimensional vector x,

$$x_j \sim \text{Cauchy}(0,1), \text{ for } j = 1, \dots, 50. \tag{3.4}$$

We call this parameterisation the "nominal" version of this model.

In addition, we also use Stan to sample from an "alternative" parameterisation of the Cauchy, based on a scale mixture of Gaussians (Vehtari et al., 2020),

$$a_j \sim \text{normal}(0,1), \qquad b_j \sim \text{Gamma}(0.5, 0.5), \qquad x_j = a_j/\sqrt{b_j}. \tag{3.5}$$

The distribution of the x vector is the same under both parameterisations, although the thin-tailed (a, b) vectors define a higher dimensional posterior that improves sampling efficiency.

In the top-left and top-middle panel of Figure 8, we show the R^* distribution for GBM and RF classifiers under both parameterisations. As shown in Vehtari et al. (2020), the nominal parameterisation results in poor sampling efficiency due to its long tails,

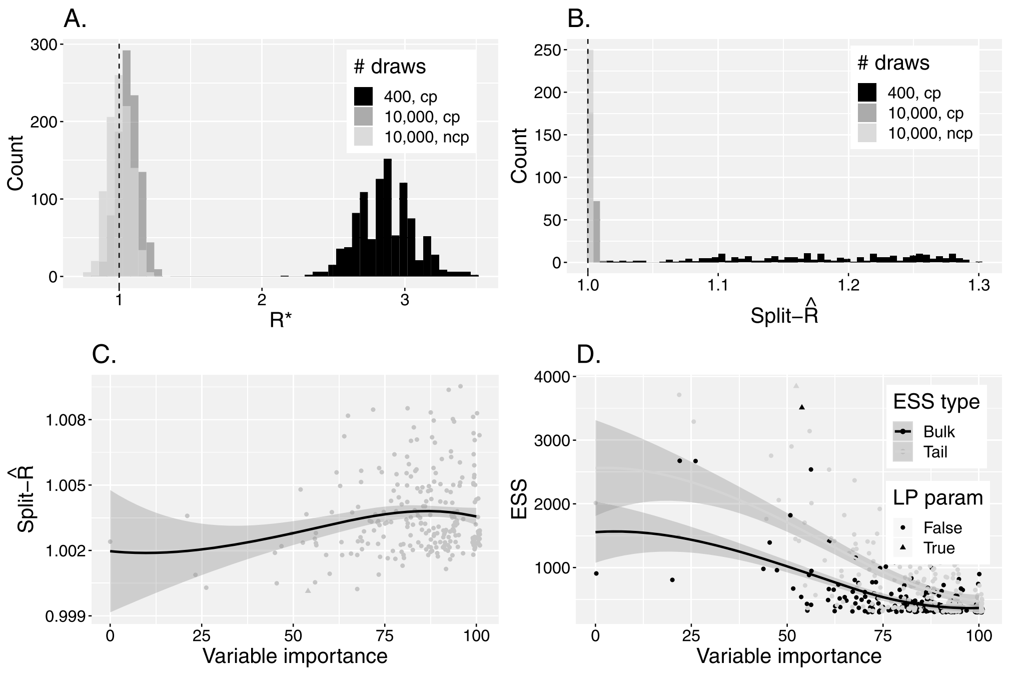

Figure 7: **Multivariate normal example with 250 dimensions.** A shows R^* distributions obtained for two MCMC samples (of differing numbers of draws: 400 and 10,000) from the centered parameterisation ("cp") and one from the non-centered version ("ncp"; with 10,000 draws); B shows the rank-normalised split-$\widehat{R}$ values for all parameters from the same MCMC runs as in A; C shows variable importance versus $\widehat{R}$ for each parameter; and D shows variable importance versus bulk- and tail-ESS as calculated by Vehtari et al. (2020). In A, 1000 R^* draws by Algorithm 2 are shown for each MCMC run. In plots C and D, horizontal jitter was added to the points and a loess fit line with standard errors overlaid.

meaning that, after 1000 MCMC post-warm-up iterations (with 1000 warm-up iterations discarded) across each of 4 chains, draws still contain information about chain identity, and, accordingly, the R^* distribution is shifted rightwards from $R^* = 1$. The alternative parameterisation fares better, and the R^* distribution is nearer $R^* = 1$, yet its mean remains above this value. In the top-right panel of Figure 8, we show the rank-normalised split-$\widehat{R}$ values across each of the 50 parameters for the same MCMC runs. The nominal parameterisation has some parameters with $\widehat{R} > 1.01$ indicating non-convergence, whereas the alternative has $\widehat{R} < 1.01$ for all parameters.

Since the R^* distribution indicated non-convergence for both parameterisations, we ran each model for sixty-times as long, although thinned by a factor of 3, resulting in 10,000 post-warm-up iterations across each of 4 chains. In the bottom row of Figure 8, we show the results for these longer runs. In these, the alternative parameterisation now has

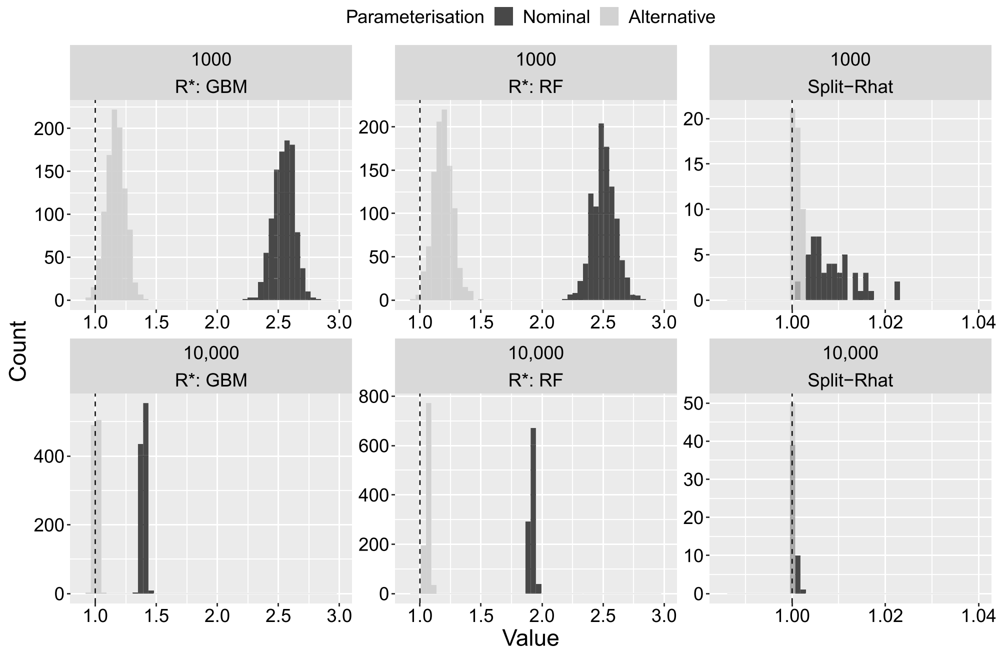

Figure 8: **Cauchy example.** Rows show convergence results for MCMC runs with 1000 (top) and 10,000 (bottom; obtained by thinning iterations by a factor of 3) post-warm-up iterations (each with half iterations discarded as warm-up) for each of 4 chains. Columns show the R^* distributions from GBM (left) and RF (middle) and rank-normalised split-$\widehat{R}$ values across all parameters (right). Shadings indicate different model paramerisations as indicated in legend.

an R^* distribution centred on $R^* = 1$ for the GBM classifier, although the RF classifier R^* distribution remains slightly rightwards of this target indicating that convergence is nearer but more iterations are likely still required. Despite the added iterations, the R^* distribution from the nominal model remains stubbornly away from 1. The $\widehat{R}$ values are all below 1.01 indicating convergence in both cases.

3.3.1 Measuring convergence objectively

To illustrate that R^* provides a reliable metric for capturing convergence, we now calculate a quantitative measure that captures how closely a sampling distribution matches the target. One measure of distributional "closeness" is the KL-divergence, which, in this case, could be used to measure the divergence from target to sampling distribution: if the target distribution is known, fitting a kernel density estimator (KDE) to samples allows an approximate (typically univariate) measure of KL-divergence to be calculated for each dimension. The trouble is, for distributions like the Cauchy with fat tails, fitting a KDE to the samples provides a noisy measure of the sampling distribution in the tails. This means that approximate KL-divergence is unreliable for these types of model. We decided not to use the Kolmogorov-Smirnov (KS) test, since it is most sensitive to differences between distributions around the median, whereas, here, we are interested in behaviour in the tails. Additionally, we found that the Anderson-Darling and Cramér-Von Mises tests (Faraway et al., 2019), which do not suffer the same shortcomings as the KS, behaved equally erratically and provided measures that were hard to intuit. The Wasserstein distance was also trialled but had great uncertainty due to the long-tails of the Cauchy. Instead, we chose a measure of distributional discrepancy based around similarity between target quantiles and sample-estimated equivalents. Specifically, we calculate the R^2 for the linear regression of actual quantile values on sample-estimated quantiles, where, if $R^2 \sim 1$, the sampling distribution recapitulates well the target quantities. In our example, we consider all percentiles: $0.1\%, 0.2\%, \ldots, 99.8\%, 99.9\%$ and calculate the mean R^2 across all 50 dimensions.

In Figure 9A, we plot this *quantile-*R^2 as a function of MCMC sample size for both parameterisations of the Cauchy model. This shows that after c.10,000 iterations, the alternative parameterisation approaches $R^2 \approx 1$; at the same number of iterations, the nominal parameterisation still provides a poor measure of tail quantiles. Next, in Figures 9B&C, we plot two measures of $\widehat{R}$, each calculated from splitting the 4 original chains into two equal halves. The first of these measures is the rank-normalised $\widehat{R}$ (Vehtari et al., 2020), which provides a separate measurement for each target dimension; in Figure 9B, we show how the maximum of this measurement across all 50 dimensions changes with sample size. After c.500 iterations, the alternative parameterisation achieves $\widehat{R} < 1.01$ for all target dimensions, and, after c.10,000 iterations, the nominal model achieves the same maximum $\widehat{R}$ value: in both cases, these suggest convergence. The second measure is multivariate $\widehat{R}$ (Brooks and Gelman, 1998), which, like R^*, yields a single measurement across all dimensions; Figure 9C shows how this metric changes with sample size for both Cauchy model parameterisations. After c.1800 iterations, multivariate $\widehat{R} < 1.01$ for the alternative parameterisation, whilst after 25,000 iterations, multivariate $\widehat{R} > 1.07$ for the nominal parameterisation indicating more draws are needed. In Figure 9D, we plot R^* against iteration for both parameterisations and for both GBM and RF classifiers: these indicate that, after 25,000 iterations, for the alternative model, $R^* \approx 1.05$ for the GBM classifier, and $R^* \approx 1.74$ for the RF classifier; for the nominal model, $R^* > 2$ for the GBM classifier $R^* > 3$ for the RF classifier: all these R^* values suggest lack of convergence. Finally, in Figures 9E&F, we plot the minimum across all the dimensions of tail- and bulk-ESS calculated as described in Vehtari et al. (2020). After c.180 iterations, the alternative parameterisation surpassed a tail-ESS of 400; after c.18,700, the nominal parameterisation did the same. Both models were quicker to pass 400 bulk-ESSs.

Comparing our measure of convergence that requires knowing the actual target distribution (*quantile-*R^2; in Figure 9A), with the various heuristic measures, all show a similar pattern: as sample size increases, the various statistics tend towards convergence. The rate at which these converge differs though, and R^* (Figure 9D) appears at least, qualitatively, most similar to *quantile-*R^2.

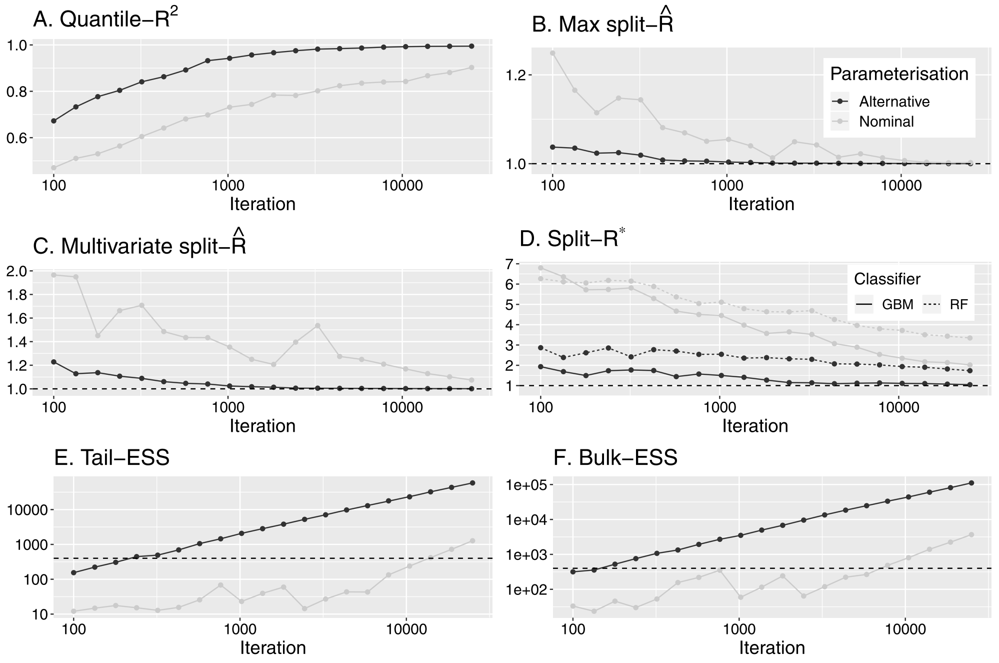

Figure 9: **Measuring convergence for the Cauchy model.** A shows a measure of convergence, the mean quantile R^2, that requires knowing the target distribution; B shows the maximum value of split-$\widehat{R}$ across each of the 50 dimensions of the target; C shows the multivariate split-$\widehat{R}$ value; D shows the value of split-R^* as calculated by Algorithm 1 for both the GBM and RF classifiers; and E and F show tail- and bulk-ESS. Horizontal dashed lines indicate recommended thresholds for each convergence statistic.

3.4 Hierarchical model: Eight schools model

We now examine a classic example used to highlight difficulties in performing inference for hierarchical models: referred to as the “Eight schools” model (see Section 5.5 in Gelman et al., 2013), which aimed to determine the effects of coaching on SAT scores in eight schools.

The model can be parameterised in two ways, as described in Vehtari et al. (2020) (and introduced in Van Dyk and Meng, 2001). The simplest way is referred to as the “centered” parameterisation and exactly mirrors the underlying statistical model,

$$\theta_j \sim \text{normal}(\mu, \tau),$$
$$y_j \sim \text{normal}(\theta_j, \sigma_j).$$

The “non-centered” parameterisation (first introduced in Van Dyk and Meng, 2001) recodes this model in a way that does not affect the joint distribution of $(\theta, \mu, \tau, \sigma)$ but makes it easier to sample from it, by introducing auxillary variables, $\tilde{\theta}_j$. This can be written as,

$$\begin{aligned}
\tilde{\theta}_j &\sim \text{normal}(0, 1),\\
\theta_j &= \mu + \tau\tilde{\theta}_j,\\
y_j &\sim \text{normal}(\theta_j, \sigma_j).
\end{aligned}$$

In both cases, θ_j are the treatment effects in the eight schools, and (μ, τ) represent the population mean and standard deviation of the distribution of these effects. In the centered parameterization, the θ_j are parameters, whereas in the non-centered parameterization, the $\tilde{\theta}_j$ are parameters and θ_j is a derived quantity.

We first used Stan (Carpenter et al., 2017) to sample from the centered model using 4 chains. Like Vehtari et al. (2020), we used settings that reduce the chance of divergent iterations for the dynamic HMC algorithm (Betancourt, 2017) (called using the "NUTS" option in Stan), meaning that the resultant sampling distribution is likely to be biased. We also used the same algorithm settings to sample from the non-centered model.

To see how R^* performed on this example, we first split each of the (post-warm-up) chains in two, as is done by default in Stan (Carpenter et al., 2017) and in Vehtari et al. (2020), resulting in 500 iterations across 8 chains. Following the same approach as in Algorithm 2, we generated R^* distributions for both the centered and non-centered models using a GBM classifier. The resultant distributions for R^* are shown in Figure 10A. In this plot, the centered model is close to convergence, whereas the non-centered is not.

In addition, to illustrate the power of R^*, we also repeat the analysis but, this time, do not split the chains in two. The results are shown in Figure 10B. In this case, because the unsplit chains do not mix with themselves, it is harder to accurately predict the chain that generated each draw, meaning that the centered model R^* values are shifted leftwards. Despite this, however, the centered model distribution for R^* still does not strongly overlap with $R^* = 1$, indicating that the model has not converged, contrasting with the non-centered model which appears near convergence.

Fig. S2 shows the equivalent of Figure 10 except using a RF classifier. The results are similar, although the R^* distributions are shifted slightly rightwards: indicating, for example, that more than 2000 draws from the non-centered model may be required for convergence.

It is recommended that $\widehat{R}$, like R^*, be calculated using split chains. In Figure 10C, we plot $\widehat{R}$ values obtained when using the original 4 chains (horizontal axis) versus those when using the split chains (vertical axis) for the ten parameters in this model; we do this for both the centered and non-centered models. These show that the values of $\widehat{R}$ for the centered model using the unsplit chains were below 1.01; when using the chains split into two halves, $\widehat{R} > 1.01$ for all but a single parameter. All parameters for the non-centered models were below 1.01 indicating convergence.

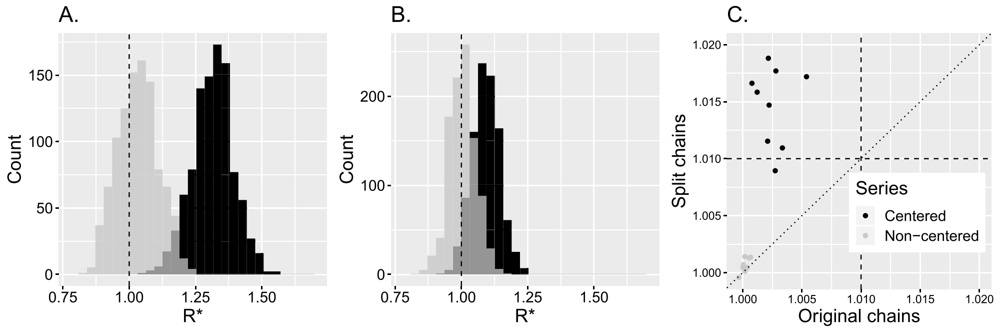

Figure 10: **Eight schools example:** R^* **distributions.** A shows draws from the R^* distribution when splitting chains in two (resulting in 8 chains); B shows the same but using the 4 original chains; C shows rank-normalised $\widehat{R}$ for the original 4 chains versus those for the 8 chains case for all ten parameters defined by the centered model – in this case, we plot horizontal and vertical dashed lines to illustrate the $\widehat{R} = 1.01$ cutoff and a $y = x$ line. The legend inset in panel C provides a key for all panels. The MCMC samples comprised 2000 draws in all cases with 1000 used as post-warm-up iterations. In panels A and B, the plots show 1000 R^* draws using Algorithm 2 using a GBM classifier for each parameterisation.

3.4.1 Understanding chain classification

To probe the predictive power of the ML classifier, we investigated how predictive accuracy varies across parameter space. After fitting the GBM model, we group MCMC draws in the test set into deciles and draw from the R^* distribution for each decile. In Figure 11, we show the results of this exercise for (A) μ and (B) τ. In the left-hand column of this figure, we show the path of four MCMC chains (here we did not split chains when calculating R^* to simplify visualisations) across the quantiles of each parameter space. To the right of each trace plot, we show the marginal distributions for each chain. In the right-hand column, we show 40 R^* draws for each decile, which were generated according to Algorithm 2 using a GBM fit to all draws. In essence, the left-hand panels explain the variation in R^* in the right-hand panels: if chains become stuck in regions of parameter space, this causes differences between the marginal distributions of the chains; these differences, in turn, allow a ML model to predict the generative chain in those same sticky regions. For example, for μ, the purple chain became stuck around the middle quantile, forcing a difference in its marginal distribution in that region, which resulted in $R^* > 1$ for the corresponding decile. Similarly, for τ, the purple chain became stuck in the lowest quantiles, elevating its marginal distribution there and resulting in improved predictive accuracy.

Figure 11 also indicates a potential limitation of R^*: namely, that as chains are progressively thinned, those regions where chains behave most idiosyncratically can be missed resulting in a reduction in classification accuracy and falsely concluding that convergence has occurred.

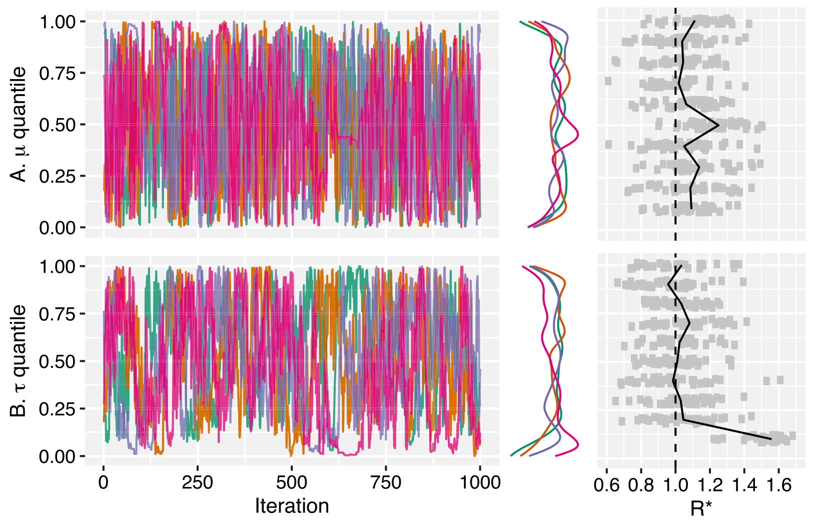

Figure 11: **Eight schools example: quantile R^* plots.** Row A shows plots for μ; row B for τ. In each row, we show the path of the four individual chains above and 40 R^* draws obtained using Algorithm 2 for each parameter quantile below. To the right of each trace plot, we show the marginal distribution of each chain estimated via kernel density estimation using Gaussian kernels. Note that, in the right-hand plots, jitter has been added to the data points.

3.5 Further experiments

Alongside the examples included in the main text, there are a number of supplementary text examples, which we briefly outline here (Lambert and Vehtari, 2020).

In §S3, we illustrate how R^* can provide a reasonable measure of convergence when the number of dimensions of a distribution is comparable to the number of draws. Specifically, this was to test that classification didn't become prone to overfitting in this limit. To test this hypothesis, we investigated two scenarios using a multivariate normal target: one with a 250-multivariate normal with high correlation between dimensions using 250 post-warm-up iterations; and another normal with 10,000 independent dimensions using up to 500 post-warm-up iterations. In both cases, sampling was done using Stan's NUTS algorithm. In both cases, R^* and rank-normalised split-$\widehat{R}$ reached similar conclusions about convergence: namely, that more iterations were needed in all experiments considered. Overall, these experiments show that R^* is a conservative convergence measure that will tend to diagnose unconvergence when there are insufficient draws.

In §S4, we illustrate the importance of splitting chains before calculating R^* to ensure poor within-chain convergence is diagnosed. We illustrate this via four examples: (a) sampling from a univariate normal and adding a linear trend over sampling time, to ensure that the sampling distributions were non-stationary; (b), similar to (a) but across a range of target distribution dimensions where only a single dimension had a non-stationary mean; (c), a bivariate normal with a non-stationary covariance; and (d), an autocorrelated sampling distribution with a univariate normal target with a range of different autocorrelations. The results of (a) echoed those presented in Vehtari et al. (2020) for $\widehat{R}$ and showed that R^* is insensitive to sampling non-convergence if it occurs within chains; splitting chains into two halves alleviates this issue. The results of (b) show that R^* calculated on split chains is able to diagnose non-stationarity in mean in a single dimension in a way that did not diminish as the numbers of dimensions considered increased. Example (c) showed that split-R^* opposed to split-$\widehat{R}$ is able to diagnose non-stationary covariance between dimensions of a target distribution. In (d), we show that R^* is able to differentiate between distributions with non-stationary target distributions and stationary ones. It also shows that R^* still functions reasonably at higher levels of chain persistence: yielding a conservative convergence measure when there are insufficient draws.

In §S5, we show that R^* performs well for two Bayesian logistic regression problems with highly multimodal posteriors. Each of these models have 1000s of parameters, and we found that it was slow to compute both $\widehat{R}$ and R^* for them. That said, the computational time for calculating $\widehat{R}$ was considerably less than was needed for R^*.

In §S6, we evaluate R^* on univariate discrete examples: one with four states (in §S6.1); another, with a larger state-space consisting of 20 states (in §S6.2). In these examples, we use a discrete Markov model to generate draws from a given target. The small and larger state-space cases show that R^* behaves as expected: given a sufficient sample size, it is able to detect differences in the transition probability matrix between chains that result in differences in the target distribution.

In §S7, we investigate how two decisions about classifiers — which classifier to use (in §S7.1) and what hyperparameters to use for it (in §S7.2) — affect calculation of R^*. In §S7.1, we test a range of popular classifiers: GBMs, RFs, k-nearest-neighbour models, support vector machines and generalised linear models across examples. This indicated that GBMs and RFs consistently had the highest classification accuracy across the examples: in higher dimensional problems, RFs tended to best GBMs. In §S7.2, we show how these two best classifiers — GBMs and RFs — depend on their hyperparameters. Across the examples we test, GBMs are more sensitive to hyperparameter choices than are RFs. Additionally, GBMs require typically more rounds of boosting in higher dimensional problems leading to more extensive algorithm runtime. Because of this, we suggest fixing GBM's hyperparameters to values that result in reasonable performance and reasonable runtime. For RFs, we suggest using a heuristic for choosing its hyperparameters that was derived in a previous empirical evaluation of RFs (Bernard et al., 2009).

In §S8, we use a slew of examples to compare R^* calculated using GBM and RF classifiers using our suggested default hyperparameter sets (given in §S2). In §S8.1, we compare how both methods are able to diagnose lack of convergence in a joint distribution (using a multivariate normal target). In §S8.2, we compare the ability of both approaches to diagnose differences in the tails of the marginal distributions between chains (using Student-t targets). In both examples, draws are generated by independent sampling meaning that an optimal R^* can be calculated based on the Bayes optimal classifier (see, for example, (Devroye et al., 2013) and §S8 for more information). Collectively, the results of §S8.1 and §S8.2 suggest that both GBM and RF classifiers can detect differences in sampling distributions, should they exist, between chains. The classification rates achieved by these two approaches exhibited similar trends to the optimal classifier, albeit with lower predictive accuracy. In higher dimensions, it is likely that the difference between optimal classification rates and those from the GBM or RF will increase: particularly, when searching for between-chain differences in tail fatness. These results also suggest a different region of optimality for each classifier: GBMs tend to perform best for low dimensional targets and RFs for moderate-high ones. This is a function of the different rules used to set the hyperparameters of each classifier (see §S2 and §S7.2): for GBMs to perform well in higher dimensions, they need more rounds of tree boosting which substantially increases training time. Because of this, we fixed the hyperparameters of the GBM to values that yield reasonable computation time. RFs are less sensitive to hyperparameter variation (§S7.2), and useful heuristics for adapting these with target dimensions are known, which we follow here, as described in §S2. Despite this dynamic choice of hyperparameters for RFs, its runtime remained reasonable over the range of examples we test in this paper.

4 Discussion

If an MCMC sampler has converged on the target distribution, the chains must be well-“mixed”, that is, given a draw, it should be impossible to discern which chain generated it. Based on this observation, we used supervised machine learning (ML) classifiers to quantify the information about the generative chain identity contained in draws. By taking the ratio of model predictive accuracy obtained on an independent test set to the accuracy of a null model (which predicts a chain’s identity uniformly at random), this defines our R^* statistic. By extracting classifier-predicted chain probabilities from each prediction in the test set, we can additionally generate an uncertainty distribution for R^*. Across a range of previously published examples, R^* was shown to be predictive of whether chains had converged.

The predominant methods for diagnosing MCMC convergence rely heavily on looking for between-chain differences in the marginal distributions along each dimension of the target. R^* naturally includes this information in building a model capable of predicting the chain that generated each draw. It also naturally includes information about the joint distribution across all dimensions of the target. Since converged chains should have similar joint distributions (implying similar marginals), any measure of convergence should account for both of these aspects. Indeed, in §3.2, we show that more established measures may indicate convergence whereas R^* shows otherwise. This indicates the complementarity of R^* to existing measures.

Different target distributions present different challenges to sampling. Because of this, there is not a unique optimal ML classifier across all cases: this is just a manifestation of the no free lunch theorem (Wolpert and Macready, 1997). Across a range of examples we tested (see §S7.1), GBM and RF classifiers performed consistently well, and we then used these across all other illustrative examples. It is possible — indeed, likely — that another ML model may exist or be invented that consistently outperforms both these classifiers. We do not see this as a problem: across the examples we considered, R^* calculated using both classifiers tended to provide a measure of convergence as or more stringent than existing diagnostics. In that sense, it is a step in the right direction. If a better classifier is found, the same apparatus we develop here can be used, and this will present a harsher test of convergence.

A different question is, "When is such a measure of convergence of practical use?". In our examples, R^* is able to diagnose poor convergence in the tails of marginal distributions: likely of practical relevance for many applications that require tail quantiles. R^* is also able to diagnose lack of convergence in the joint distribution — even if the marginals appear converged. Indeed, it is less clear how the joint distribution can be unconverged when the marginals appear so, but we found examples where this appeared true. A consequence of poor convergence of the joint distribution would be for prediction and, by corollary, model comparison. Further work examining these consequences further is needed, and R^* can help to identify and monitor fruitful candidate systems.

In §S5, we fit Bayesian models with many 1000s of parameters then used R^* to diagnose convergence, finding that R^* was considerably more expensive to calculate than $\widehat{R}$. The time complexity of training RFs is thought to be $\mathcal{O}(m_{\text{try}} n_{\text{tree}} n_{\text{data}} \log\, n_{\text{data}})$ (Louppe, 2014, chapter 5), and given the similarities with GBMs (which also builds many decision trees), it is likely to be similar. If so, this suggests that larger statistical models (usually needing more MCMC iterations) may currently be beyond the reach of R^*. That said, it is possible to reduce the runtime for R^* using thinned draws (although this risks losing chain idiosyncracies) and using a subset of dimensions (although this risks losing problematic dimensions). Indeed, in §3.2.2, §3.3 and §S5, we use these strategies and, nonetheless, find that R^* provides a stringent measure of convergence.

Many implementations of $\widehat{R}$ suggest splitting chains in two before calculating it. In a number of examples, we trial this before calculating R^* and find that this approach leads to more accurate chain prediction. We recommend that this practice be adopted whenever R^* is calculated to ensure that this measure is maximised. Additionally, our non-parametric calculation method for R^* makes it possible to include any covariates which may be useful features for prediction, such as an "iteration block" indicator variable taking values $1, 2, \ldots, K$ in each of K blocks of contiguous iterations. If each chain is thoroughly mixed with itself, including this additional information shouldn't change R^*; by contrast, if the chains are random walk-like, this information should boost R^*.

MCMC enables inference across a wide range of models encountered across the social, biological and physical sciences. Its ease of implementation, however, masks important underlying fragilities in the method. Namely, that unless the chains have converged to a truly stationary distribution, the draws generated are not faithful depictions of the posterior. In this paper, we introduce a new metric, R^*, that is especially good at diagnosing poor convergence in the joint sampling distribution – an area that has received insufficient attention thus far. R^* can straightforwardly be introduced into existing MCMC libraries and could provide a measure of convergence complementary to existing metrics.

Supplementary Material

Supplementary materials: R^* convergence diagnostic. (DOI: 10.1214/20-BA1252SUPP; .pdf). Further experiments using R^*.

References

Bernard, S., Heutte, L., and Adam, S. (2009). "Influence of hyperparameters on random forest accuracy." In *International Workshop on Multiple Classifier Systems*, 171–180. Springer. 355, 358, 374

Betancourt, M. (2017). "A conceptual introduction to Hamiltonian Monte Carlo." *arXiv preprint arXiv:1701.02434*. MR1699395. doi: `https://doi.org/10.1017/CBO9780511470813.003`. 353, 365, 371

Bingham, E., Chen, J., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. (2019). "Pyro: Deep universal probabilistic programming." *The Journal of Machine Learning Research*, 20(1): 973–978. 353

Boehmke, B. and Greenwell, B. (2019). *Hands-on machine learning with R*. CRC Press. 355, 357, 358

Breiman, L. (2001). "Random forests." *Machine Learning*, 45(1): 5–32. MR3874153. 355, 356

Brooks, S., Gelman, A., Jones, G., and Meng, X. (2011). *Handbook of Markov chain Monte Carlo*. CRC Press. MR2742422. doi: `https://doi.org/10.1201/b10905`. 354

Brooks, S. P. and Gelman, A. (1998). "General methods for monitoring convergence of iterative simulations." *Journal of Computational and Graphical Statistics*, 7(4): 434–455. MR1665662. doi: `https://doi.org/10.2307/1390675`. 369

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., and Riddell, A. (2017). "Stan: A probabilistic programming language." *Journal of Statistical Software*, 76(1). 353, 354, 371

Devroye, L., Györfi, L., and Lugosi, G. (2013). *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media. MR1383093. doi: `https://doi.org/10.1007/978-1-4612-0711-5`. 375

Dillon, J., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. (2017). "Tensorflow distributions." *arXiv preprint arXiv:1711.10604*. 353

Faraway, J., Marsaglia, G., Marsaglia, J., and Baddeley, A. (2019). *goftest: Classical*

Goodness-of-Fit Tests for Univariate Distributions. R package version 1.2-2. URL `https://CRAN.R-project.org/package=goftest` 369

Friedman, J. (2001). "Greedy function approximation: a gradient boosting machine." *Annals of Statistics*, 1189–1232. MR1873328. doi: `https://doi.org/10.1214/aos/1013203451`. 355, 356, 366

Ge, H., Xu, K., and Ghahramani, Z. (2018). "Turing: A language for flexible probabilistic inference." 353

Gelman, A. and Rubin, D. (1992a). "Inference from iterative simulation using multiple sequences." *Statistical Science*, 7(4): 457–472. 354

Gelman, A. and Rubin, D. (1992b). "A single series from the Gibbs sampler provides a false sense of security." *Bayesian Statistics*, 4: 625–631. 354

Gelman, A., Stern, H., Carlin, J., Dunson, D., Vehtari, A., and Rubin, D. (2013). *Bayesian data analysis*. Chapman and Hall/CRC. MR3235677. 354, 370

Geman, S. and Geman, D. (1984). "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6): 721–741. 353

Greenwell, B., Boehmke, B., Cunningham, J., Developers, G., and Greenwell, M. B. (2019). "Package 'gbm'." 355, 357, 366

Hoffman, M. and Gelman, A. (2014). "The No-U-turn Sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *Journal of Machine Learning Research*, 15(1): 1593–1623. MR3214779. 353, 365

Kuhn, M. et al. (2008). "Building predictive models in R using the Caret package." *Journal of Statistical Software*, 28(5): 1–26. 357

Lambert, B. (2018a). *A student's guide to Bayesian statistics*. Sage Publications Ltd. 354

Lambert, B. (2018b). "YouTube video: Bob's bees: the importance of using multiple bees (chains) to judge MCMC convergence." 354

Lambert, B. and Vehtari, B. (2020). "Supplementary Material of "R^*: A Robust MCMC Convergence Diagnostic with Uncertainty Using Decision Tree Classifiers"." *Bayesian Analysis*. doi: `https://doi.org/10.1214/20-BA1252SUPP`. 373

Liaw, A. and Wiener, M. (2002). "Classification and regression by randomForest." *R News*, 2(3): 18–22. 357

Louppe, G. (2014). "Understanding random forests: From theory to practice." *arXiv preprint arXiv:1407.7502*. 376

Lunn, D., Thomas, A., Best, N., and Spiegelhalter, D. (2000). "WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility." *Statistics and Computing*, 10(4): 325–337. 353

Neal, R. et al. (2011). "MCMC using Hamiltonian dynamics." *Handbook of Markov chain Monte Carlo*, 2(11): 2. MR2858447. 353

Plummer, M. et al. (2003). "JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling." In *Proceedings of the 3rd international workshop on Distributed Statistical Computing*, volume 124. Vienna, Austria. 353

Salvatier, J., Wiecki, T., and Fonnesbeck, C. (2016). "Probabilistic programming in Python using PyMC3." *PeerJ Computer Science*, 2: e55. 353

Van Dyk, D. and Meng, X. (2001). "The art of data augmentation." *Journal of Computational and Graphical Statistics*, 10(1): 1–50. MR1936358. doi: `https://doi.org/10.1198/10618600152418584`. 370

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., and Bürkner, P. (2020). "Rank-normalization, folding, and localization: An improved R-hat for assessing convergence of MCMC." *Bayesian Analysis*. 354, 356, 359, 364, 365, 366, 367, 369, 370, 371, 374

Wolpert, D. and Macready, W. (1997). "No free lunch theorems for optimization." *IEEE Transactions on Evolutionary Computation*, 1(1): 67–82. 375

Acknowledgments

The authors would like to thank the anonymous reviewers for comments on previous drafts of the paper that lead to significant improvements. We would also like to thank Paul Bürkner and Jonah Gabry, with whom useful discussions were had during the preparation of this manuscript.