R^*: A robust MCMC convergence diagnostic with uncertainty using gradient-boosted machines

A Preprint

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

Aki Vehtari
Department of Computer Science
Aalto University
Finland
`aki.vehtari@aalto.fi`

September 1, 2020

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 finite sample performance. 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 whether 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 using all the information in the chain, including that given by the joint sampling distribution, which is currently largely overlooked by existing approaches. Since our choice of machine learning classifier, a gradient-boosted regression trees model (GBM), provides uncertainty in predictions, as a byproduct, we obtain uncertainty in R^*. The method is straightforward to implement, robust to GBM hyperparameter choice, 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) and PyMC3 (Salvatier et al., 2016) 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 in particular (Carpenter et al., 2017) is a great exemplar of this), go to great lengths to communicate to users any issues with sampling.

The most important determination of whether MCMC has worked is whether 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 predominant 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. 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, 1992). 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, 1992). 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 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 the chain that *caused* a draw. In this vein, we use a machine learning (ML) classifier to measure convergence. Specifically, we train a ML classifier to predict the chain that generated each observation. By evaluating the performance of the classifier on a held-out test set, this provides a new convergence metric. To maximise predictive accuracy, our chosen ML classifier naturally exploits differences in the full joint distributions between chains, which means it's 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 classifier we use can straightforwardly be interrogated to determine which parameters were most important for generating predictive accuracy. For our ML classifier, we use gradient-boosted regression trees (Friedman, 2001; Greenwell et al., 2019) ("GBM"), since these are known to perform well for the types of tabular data that our problem presents (Chollet and Allaire, 2018). For the types of problem we test, 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. It is also insensitive to GBM's hyperparameters and provides a measure of convergence robust to various Markov chain pathologies. In addition, since GBMs 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 introduced in Vehtari et al. (2020) and elsewhere. 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 determine to which chain a draw belongs from its value. This is possible if there are differences in the sampling distribution for any dimension, θ, in the target distribution (Fig. 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 (Fig. 2).

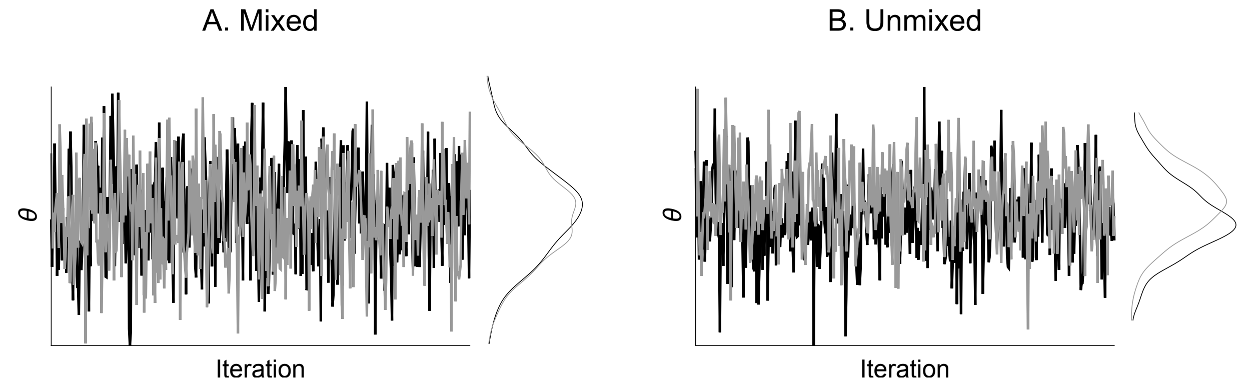

Figure 1: **Chain prediction based on the marginal distribution of a single parameter.** A shows the path of two chains that have mixed (with marginal distribution to the right of panel); B shows two chains that have not mixed.

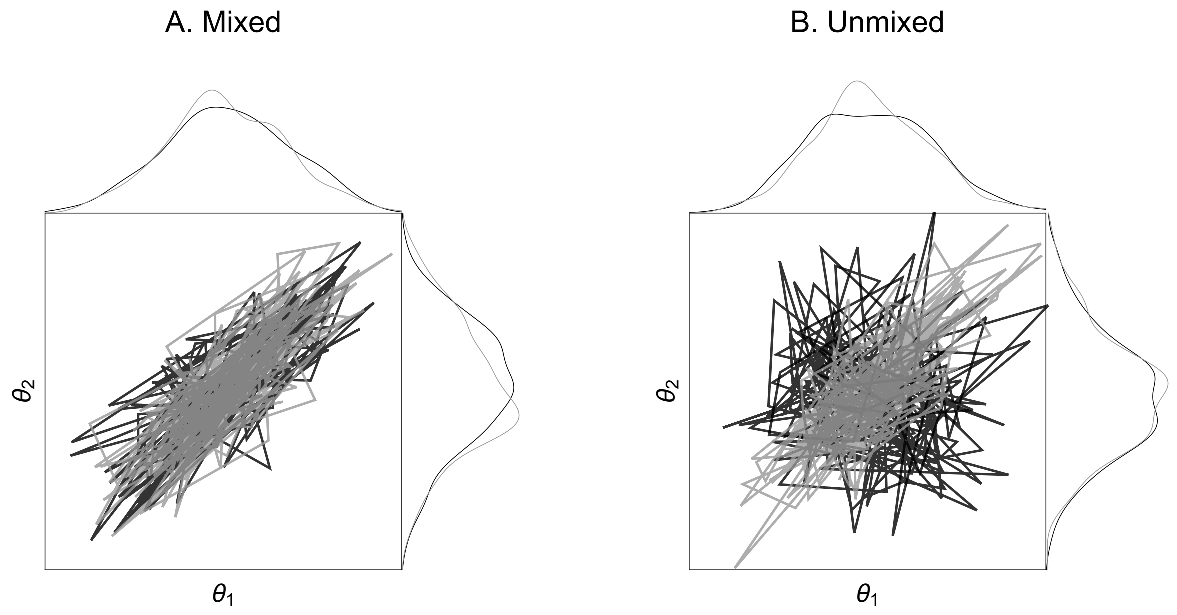

Figure 2: **Chain prediction based on the joint distribution of two parameters where each chain's marginals are the same.** A shows the path of two chains that have mixed resulting in similar sampling distributions (to the right and above each panel); B shows two chains that have not mixed.

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 machine learning (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 ML 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^*.

The ML classifier we use here is a gradient-boosted regression tree (also known as a type of gradient-boosted machine or GBM, introduced in Friedman (2001)), which experience has dictated to be a highly predictive framework for use in tabular data (Chollet and Allaire, 2018) like ours. Specifically, we use the GBM implementation in **R**'s "Caret" package

Algorithm 1 R^* calculation

```
Given chain-wise draws from the target, $\{X^{\{1\}}, X^{\{2\}}, ..., X^{\{N\}}\}$ 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 $X_{\text{train}} = (X^{\{1\}}_{\text{train}}, X^{\{2\}}_{\text{train}}, ..., X^{\{N\}}_{\text{train}})^T$
Stack $X_{\text{test}} = (X^{\{1\}}_{\text{test}}, X^{\{2\}}_{\text{test}}, ..., X^{\{N\}}_{\text{test}})^T$
Train ML model to classify chain id from any draw, $x$: 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 $R^*$
```

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, ML$(x|X_{\text{train}}) \rightarrow (p_1, p_2, ..., p_N)$:
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, $(p_1^{\{s\}}, p_2^{\{s\}}, ..., p_N^{\{s\}}) = \text{ML}(x^{\{s\}}|X_{\text{train}})$
        Draw a chain id, $c^{\{s\}} \sim \text{categorical}(p_1^{\{s\}}, p_2^{\{s\}}, ..., p_N^{\{s\}})$
        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 $(R^{*\{1\}}, R^{*\{2\}}, ..., R^{*\{I\}})$
```

(Kuhn et al., 2008), which, in turn, uses the "gbm" package (Greenwell et al., 2019). The data for each chain has 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 tests: here, we use 70% of draws for training and 30% for testing. Our GBM was rapid to execute training then prediction on the testing set (taking $\mathcal{O}$(seconds) on a desktop computer for both these steps for most models we consider in §3), and its predictive performance was insensitive to its hyperparameters (although we explore this in §3.1.5). Unless stated otherwise, in the examples explored in §3, the GBM hyperparameter settings we used were: an interaction depth of 3, a shrinkage parameter of 0.1, 10 observations being the minimum required for each node, and that 50 trees would be grown.

From the GBM 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, GBM outputs a simplex of chain probabilities: $\boldsymbol{p}^{\{s\}} = (p_1^{\{s\}}, p_2^{\{s\}}, ..., p_N^{\{s\}})$, 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 $\boldsymbol{a} = (a^{\{1\}}, a^{\{2\}}, ..., a^{\{S_{\text{test}}\}})$, whose average yields a single $R^{*\{i\}} = N\bar{a}$ estimate for iteration i. We then iterate this process, for $i = 1, 2, ..., I$, producing a set of $(R^{*\{1\}}, R^{*\{2\}}, ..., R^{*\{I\}})$, which collectively represent a distribution for R^*.

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
	Robustness of R^* to ML model hyperparameters	3.1.5
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**	S1
Non-stationary marginals	**Detecting time-varying sampling distributions**	S2
	Trends in mean across all chains	S2.1
	Trends in mean in a single dimension	S2.2
	Trends in covariance	S2.3
	Sensitivity of R^* to chain persistence	S2.4
Ovarian and prostate models	**Bayesian models with many parameters and multimodal posteriors**	S3

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

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.

3.1 Heterogeneity in chain variance: autoregressive example

In this section, we show how R^* is able to detect heterogeneous variance across Markov chains. Apart from demonstrating how R^* performs, 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); to differing numbers of chains (§3.1.4); and, finally, to the ML model hyperparameters (§3.1.5). 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,

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

where $\epsilon_t \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma)$, $\rho = 0.3$ and $t = 1, 2, ..., 2000$. 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 perform 1000 replicates where, in each case, we generate 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 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 Fig. 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 Fig. 3B, we show that $R^* > 1$ for all replicates, indicating that the chains had not converged in all cases. In Fig. 3C, we show the rank-normalised split-$\widehat{R}$ as calculated in Vehtari et al. (2020) for each replicate; as for R^*, this metric indicates the chains had not converged in all replicates (as diagnosed by $R^* < 1.01$).

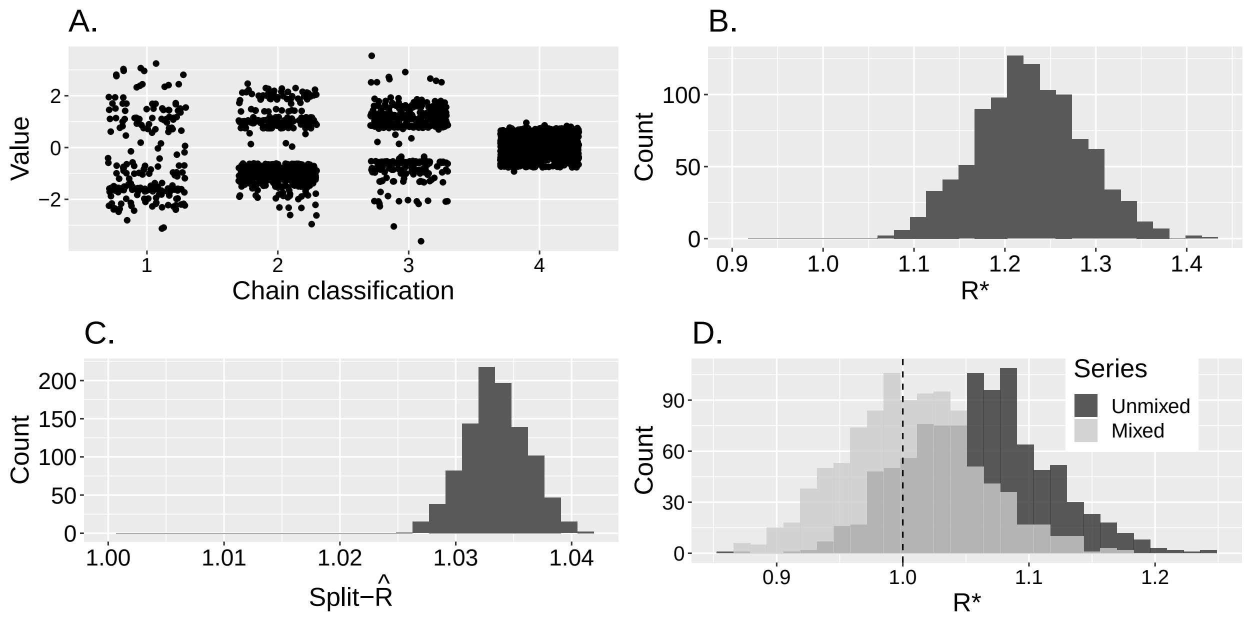

Figure 3: **Autoregressive example.** A shows how the GBM's classifications vary according to the draw's value for an example model fit; B shows R^* values generated by Algorithm 1 across 1000 replicate datasets; C shows corresponding rank-normalised split-$\widehat{R}$ values as calculated in Vehtari et al. (2020) 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" 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.

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 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 Fig. 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, since more training data leads to better ML models. 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 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.

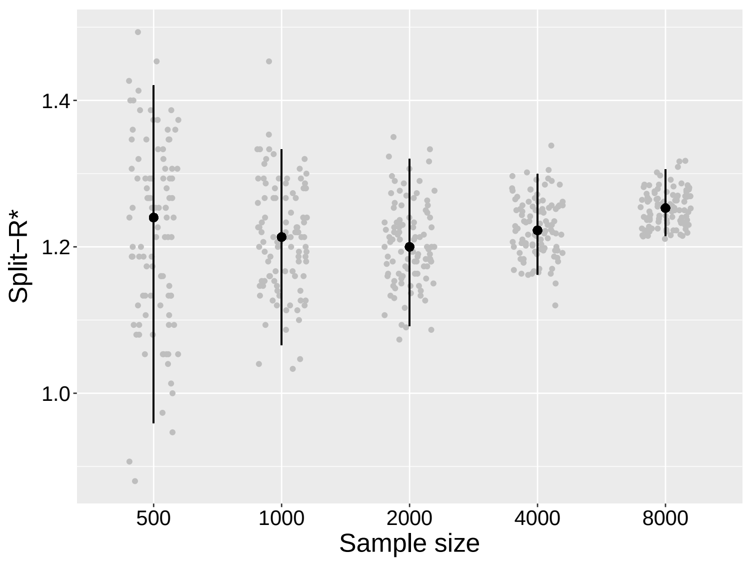

Figure 4: **Autoregressive example: R^* stochasticity.** The horizontal axis shows sample size; the vertical axis shows the value of R^* calculated as per Algorithm 1 applied to chains split into two halves. Grey points show the value of R^* for each replicate (jitter was added to point positions). Black 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.

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 Fig. 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 becomes a harder problem when there are more categories, we now demonstrate how R^* (as calculated by Algorithm 1) performs across various number 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 Fig. 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 is 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.

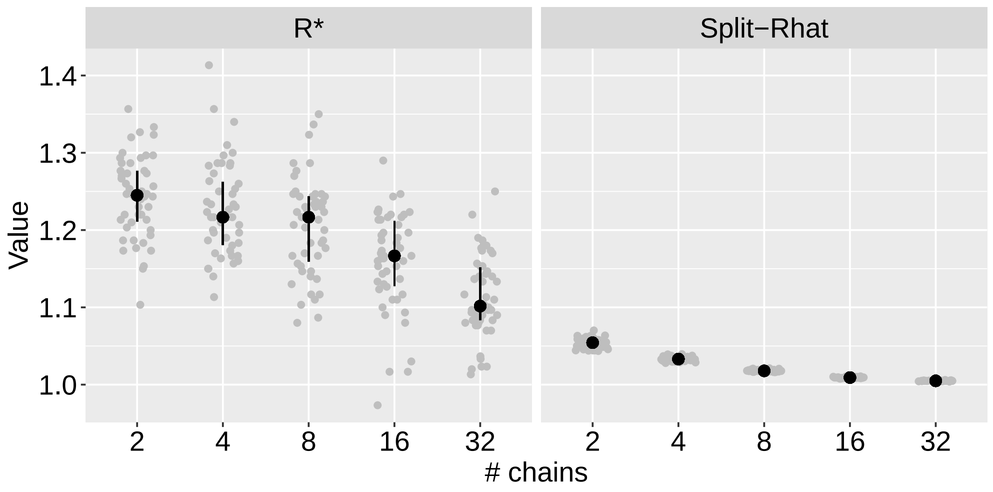

Figure 5: **Autoregressive example: sensitivity to number of chains.** The horizontal axis shows the number of chains used in the data generating process described in §3.1.4. The vertical axis shows the value of R^* as calculated by Algorithm 1 (left panel) on chains split into two halves, and rank-normalised split-$\widehat{R}$ as calculated in Vehtari et al. (2020) (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.

3.1.5 Robustness to hyperparameter choice

The performance of GBM, like all ML methods, depends on its hyperparameters. To investigate the robustness of R^* to GBM hyperparameter choice, we performed a sensitivity analysis. In this, we fit GBMs with six different sets of hyperparameters to replicates consisting of the same data used to generate Fig. 3. The hyperparameters we varied were the interaction depth (3, 5 or 7) and the number of trees (50, 100 or 200). The resultant R^* point estimates across 200 replicates for each hyperparameter setting are shown in Fig. 6. These show that hyperparameter choice affects R^*, with the highest median values obtained for an interaction depth of 3 with 50 trees (our default across all the examples we consider in this paper). This analysis does suggest, however, that this effect is relatively minor: indeed, across the various hyperparameter sets considered, the percentage of replicates where $R^* < 1$ did not strongly vary.

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.

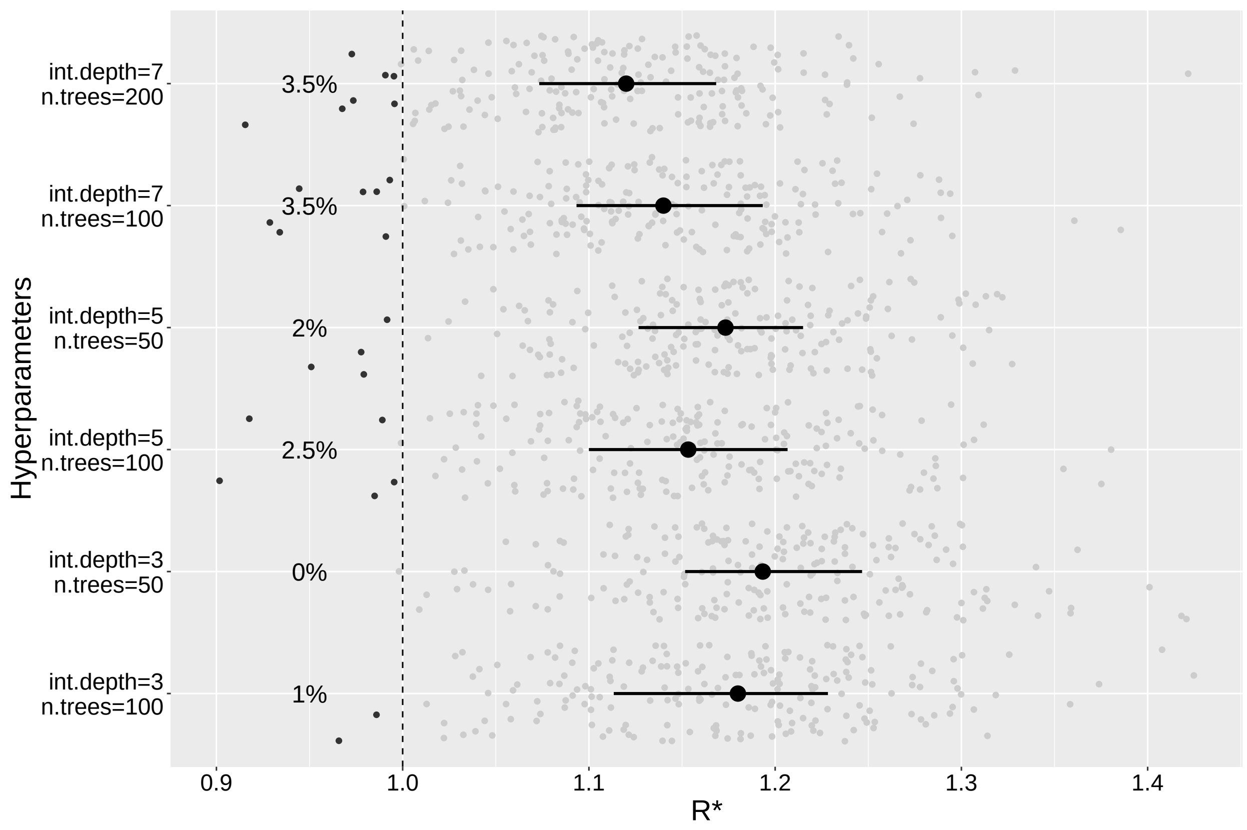

Figure 6: **Autoregressive example: sensitivity to GBM hyperparameters.** The rows correspond to GBM models fit using different hyperparameter sets (as indicated for interaction depth and number of trees and using a shrinkage of 0.1 and a minimum observations per node of 10 in all cases). The horizontal axis shows the R^* values generated using Algorithm 1 using split chains; points have jitter added and are coloured according to whether they exceed the threshold $R^* = 1$, with the percentage below this threshold annotated. The point-ranges shown indicate the 25%, 50% and 75% quantiles for each model.

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, which is shown in Fig. 7. The mean of this distribution is 1.14, and >99% of R^* draws are above 1, indicating 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

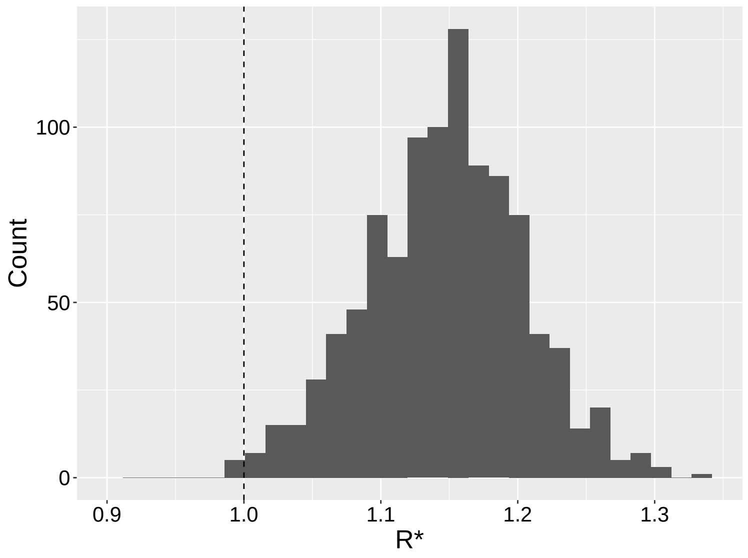

Figure 7: **Bivariate normal example.** The distribution for R^* across 1000 draws.

Stan's NUTS algorithm 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,

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{A}^{-1}), \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^{250}$. For each set of draws, we used Algorithm 2 to generate an uncertainty distribution for R^*, which is shown in Fig. 8A. 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 95% of parameters (Fig. 8B), 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,

$$\boldsymbol{A}^{-1} = \boldsymbol{L}\boldsymbol{L}^T, \qquad \boldsymbol{x} = \boldsymbol{L}\boldsymbol{z}, \qquad z_j \sim \mathcal{N}(0, 1), \text{ for } j = 1, 2, ..., 250. \tag{3}$$

where $\boldsymbol{L}$ is the Cholesky decomposition of the covariance matrix, $\boldsymbol{A}^{-1}$. Fig. 8A shows the R^* distribution resultant from 10,000 NUTS iterations in this case: now the distribution has mean $R^* = 1.00$. Fig. 8B 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)), and this 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. (2) with 10,000 MCMC iterations (thinning by a factor of 5) for each chain, we plot in Fig. 8C 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: $\rho = 0.33, S = 1763958, p < 0.01$). In Fig. 8D, 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.

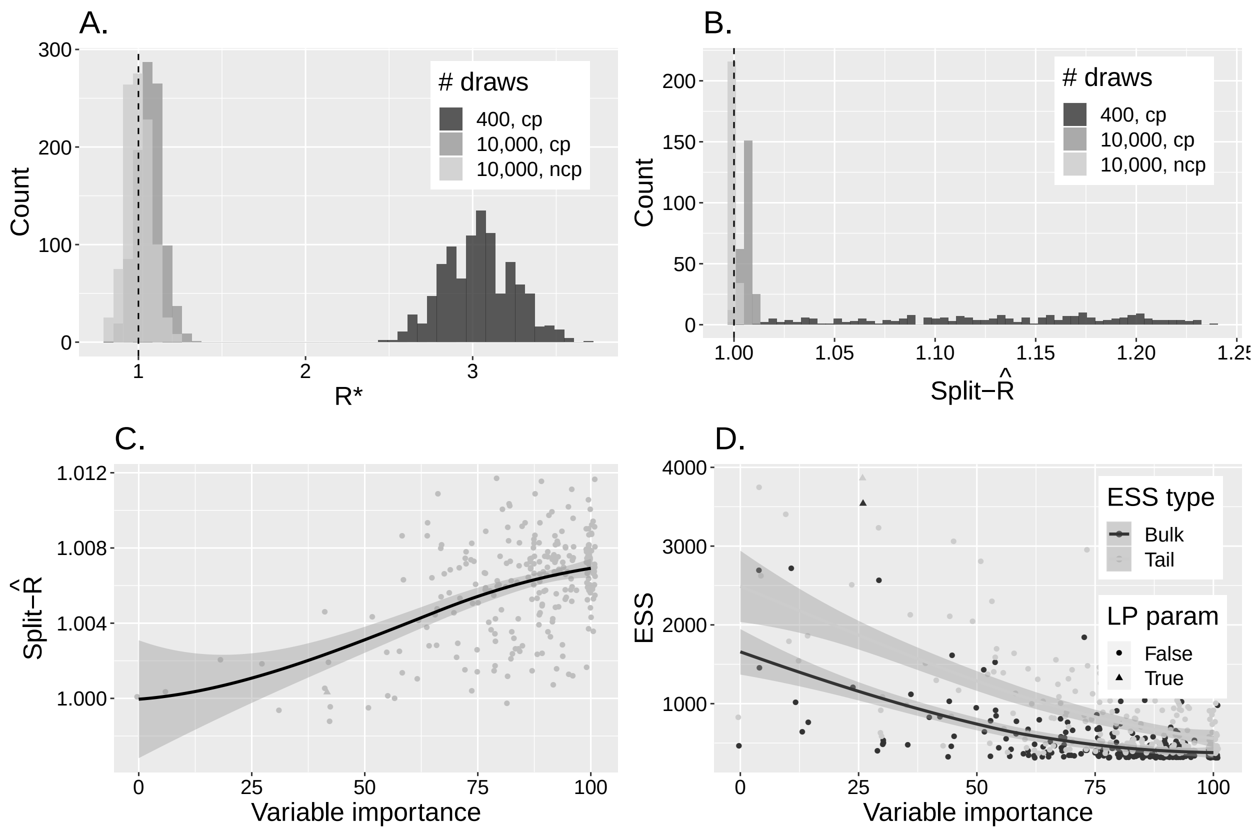

Figure 8: **Multivariate normal example with 250 dimensions.** A shows R^* distributions obtained for two MCMC samples (of differing numbers of draws: 400 and 10,000) from the centered parameterisation ("cp") and one from the non-centered version ("ncp"; with 10,000 draws); B shows the rank-normalised 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.

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,

$$x_j \sim \text{Cauchy}(0, 1), \text{ for } j = 1, ..., 50. \tag{4}$$

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),

$$a_j \sim \mathcal{N}(0, 1), \qquad b_j \sim \text{Gamma}(0.5, 0.5), \qquad x_j = a_j/\sqrt{b_j}. \tag{5}$$

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 panel of Fig. 9, we show the R^* distribution under both parameterisations. As shown in Vehtari et al. (2020), the nominal parameterisation results in poor sampling efficiency due to its long tails, 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 Fig. 9, 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 $\widehat{R} > 1.01$, indicative 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 Fig. 9, we show the results for these longer runs. In these, the alternative parameterisation now has an R^* distribution centered on $R^* = 1$ and, hence, we are more confident that convergence has occurred. 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.

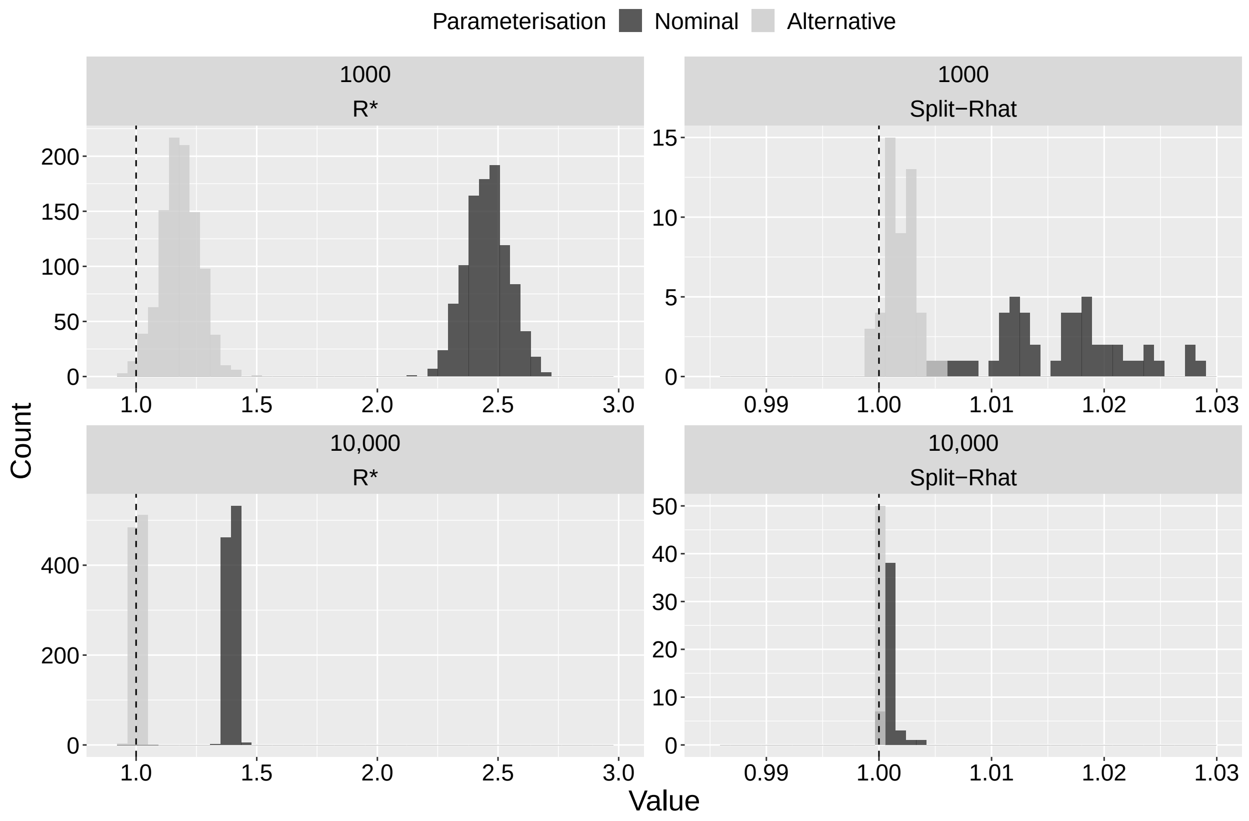

Figure 9: **Cauchy example.** Rows show convergence results for MCMC runs with 1000 (top) and 10,000 (bottom; obtained by thinning iterations by a factor of 3) post-warm-up iterations (each with half iterations discarded as warm-up) for each of 4 chains. Columns show the R^* distributions (left) and rank-normalised split-$\widehat{R}$ values across all parameters (right). Shadings indicate different model paramerisations as indicated in legend.

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%,...,99.8%, 99.9% and calculate the mean R^2 across all 50 dimensions.

In Fig. 10A, 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 Figs. 10B&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 Fig. 10B, we show how the maximum of this measurement across all 50 dimensions changes with sample size. After c.550 iterations, the alternative parameterisation achieves $\widehat{R} < 1.01$ for all target dimensions, and, after c.10,400 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; Fig. 10C 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 $\widehat{R} > 1.07$ indicating more draws are needed. In Fig. 10D, we plot R^* against iteration for both models: these indicate that, after 25,000 iterations, $R^* \approx 1.05$ for the alternative model, and $R^* > 2$ for the nominal parameterisation: both these R^* values suggest lack of convergence. Finally, in Figs. 10E&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 "omniscient" measure of convergence shown in Fig. 10A, with the various derived 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^* (Fig. 10D) appears at least, qualitatively, most similar to our "omniscient" measure.

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). The simplest way is referred to as the "centered" parameterisation and exactly mirrors the underlying statistical model,

$$\theta_j \sim \mathcal{N}(\mu, \tau),$$
$$y_j \sim \mathcal{N}(\theta_j, \sigma_j).$$

The "non-centered" parameterisation 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,

$$\tilde{\theta}_j \sim \mathcal{N}(0, 1),$$
$$\theta_j = \mu + \tau\tilde{\theta}_j,$$
$$y_j \sim \mathcal{N}(\theta_j, \sigma_j).$$

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 NUTS algorithm (Betancourt, 2017), 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. The resultant distributions for R^* are shown in Fig.11A. 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 Fig.11B. In this case, because the unsplit chains do not mix with themselves, it is harder to

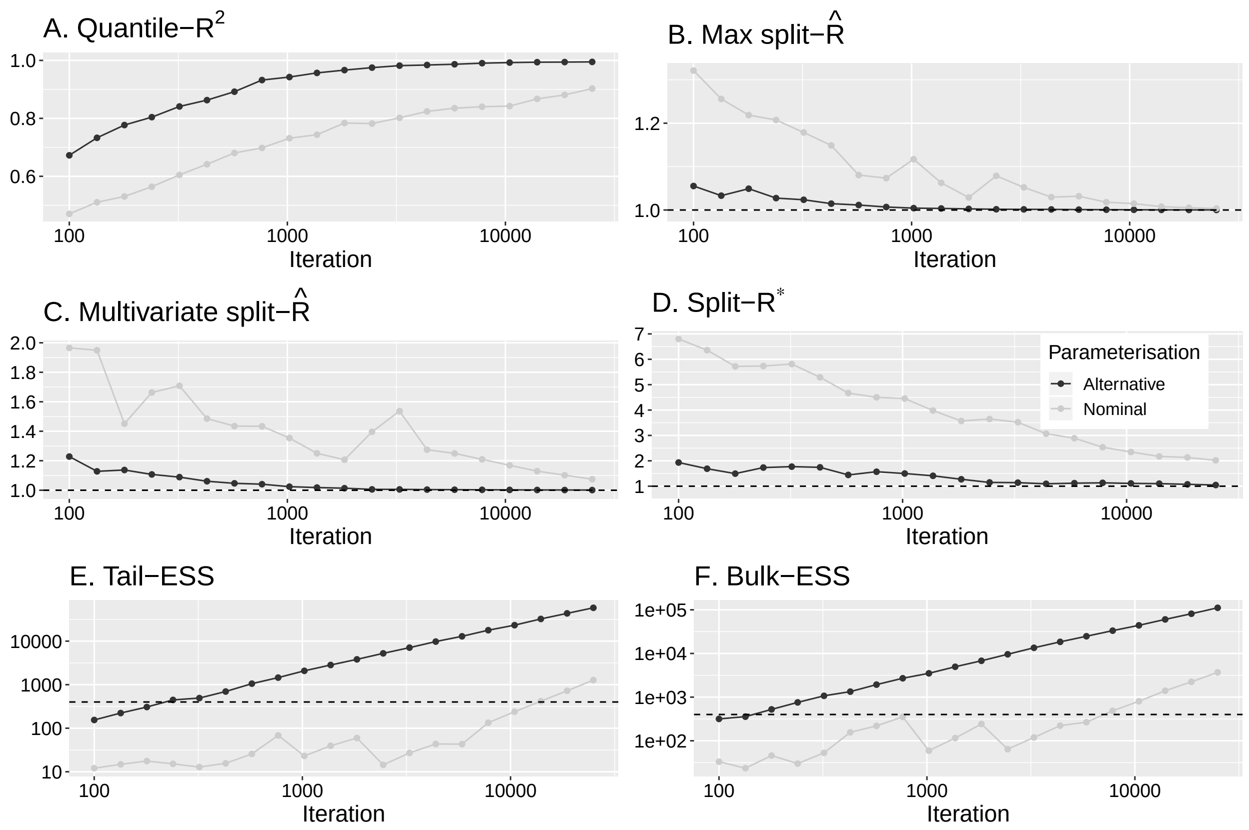

Figure 10: **Measuring convergence for the Cauchy model.** A shows a measure of convergence, the mean quantile R^2, that requires knowing the target distribution; B shows the maximum value of split-$\widehat{R}$ across each of the 50 dimensions of the target; C shows the multivariate split-$\widehat{R}$ value; D shows the value of split-R^* as calculated by Algorithm 1; and E and F show tail- and bulk-ESS. Dashed lines indicate recommended thresholds for each convergence statistic.

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.

It is recommended that $\widehat{R}$, like R^*, be calculated using split chains. In Fig. 11C, 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 for both the split and unsplit cases were, for all but a single parameter, above 1.01, echoing the results for R^*; also, like R^*, $\widehat{R}$ is higher for the split case. All parameters for the non-centered models were below 1.01, indicating convergence.

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 into deciles and draw from the R^* distribution for each decile. In Fig. 12, we show the results of this exercise for (A) μ and (B) τ. In the top row 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. Above the top panel, we show the marginal distributions for each chain. In the bottom row, 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 top rows explain the variation in R^* in the bottom 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

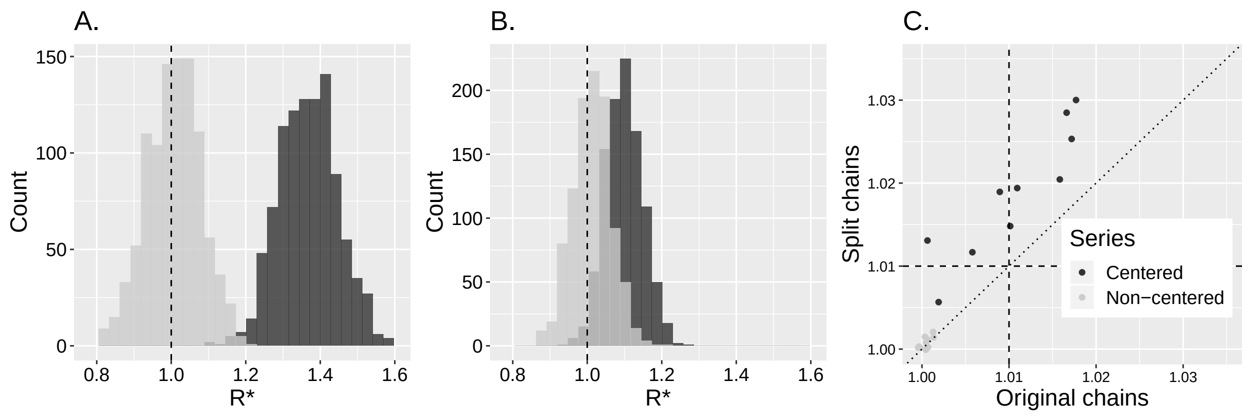

Figure 11: **Eight schools example: R^* distributions.** A shows draws from the R^* distribution when splitting chains in two (resulting in 8 chains); B shows the same but using the 4 original chains; C shows rank-normalised $\widehat{R}$ for original 4 chains versus those for the 8 chains case for all 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 for each parameterisation.

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 ML predictive accuracy.

Fig. 12 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 ML classification accuracy and falsely concluding that convergence has occurred.

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.

In §S1, 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 ML 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 §S2, 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 insufficient draws.

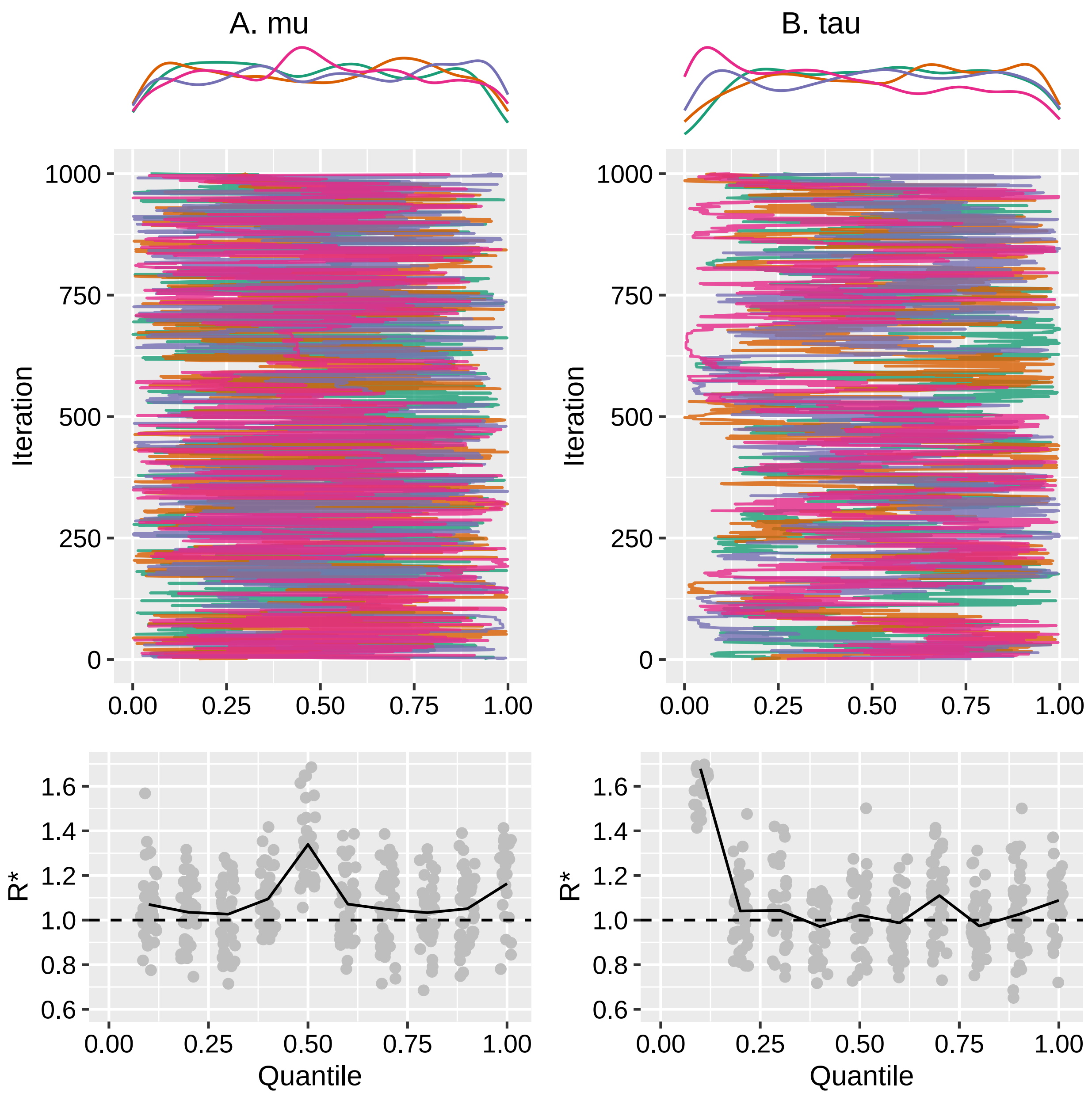

Figure 12: **Eight schools example: quantile R^* plots.** Column A shows plots for μ; Column B for τ. In each column, we show the path of the four individual chains above and 40 R^* draws obtained using Algorithm 2 for each parameter quantile below. Above the top row, we show the marginal distribution of each chain estimated via kernel density estimation using Gaussian kernels. Note that, in the bottom plots, jitter has been added to the data points.

In §S3, 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^*.

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 ML 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 ML-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 is not a sleight on existing measures, more that this illustrates the complementarity of R^* to them.

When first starting to develop R^*, it was unclear to us whether *any* metric aligned with a supervised ML method would be overly sensitive to its hyperparameters. With gradient-boosted regression tree models, we found this not to be the case. Indeed, after a little experimenting, we found that a default set of hyperparameters (which we report in §2) sufficed across all our examples: perhaps, because the classification boundaries for this task are unlikely to be very complex compared to (say) machine vision tasks. To ensure maximal predictive capability of any fitted ML model, however, it is prudent to optimise it over choice of hyperparameters, and further work is needed to determine the importance of this step.

In §S3, 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 computational cost of training GBMs has been suggested to be $\mathcal{O}(Nk)$, where N is the number of training data points and k is the number of dimensions of the target (RUser4512, 2018). If so, this suggests that larger models (usually needing more MCMC iterations) may currently be beyond the reach of R^*.

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 ML 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 $1, 2, ..., K$ 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.

5 Contributions

BL conceived of the original idea for R^*, carried out all analyses and wrote the original draft of the paper. AV reviewed the paper and made a host of recommendations: including suggesting a large range of new case studies on which to trial the method and also suggesting new visualisations for diagnosing how predictive performance depends on parameter quantiles; these collectively widened the scope of the original paper and substantially improved its quality. BL and AV reviewed and edited the draft of the paper.

References

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434*.

Brooks, S., A. Gelman, G. Jones, and X. Meng (2011). *Handbook of Markov chain Monte Carlo*. CRC press.

Brooks, S. P. and A. Gelman (1998). General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics 7*(4), 434–455.

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

Chollet, F. and J. Allaire (2018). *Deep learning with R*. Manning Publications, Shelter Island, NY.

Faraway, J., G. Marsaglia, J. Marsaglia, and A. Baddeley (2019). *goftest: Classical Goodness-of-Fit Tests for Univariate Distributions*. R package version 1.2-2.

Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 1189–1232.

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

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

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

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

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

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

Lambert, B. (2018a). *A Student's Guide to Bayesian Statistics*. Sage Publications Ltd.

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

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

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

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.

RUser4512 (2018). Computational complexity of machine learning algorithms. https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/. Accessed: 2020-08-04.

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

Vehtari, A., A. Gelman, D. Simpson, B. Carpenter, and P. Bürkner (2020). Rank-normalization, folding, and localization: An improved r-hat for assessing convergence of MCMC. *Bayesian Analysis*.