Bayesian Analysis (0000) **00**, Number 0, pp. 1

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

 DOI: 0000

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 (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).

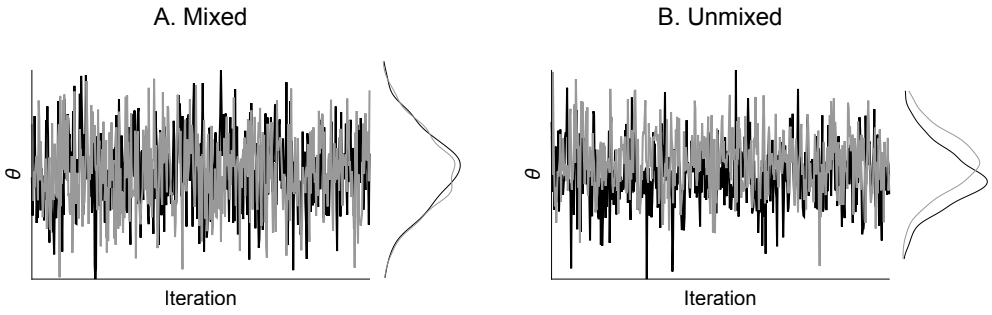

Figure 1: **Chain prediction based on the marginal distribution of a single parameter.** A shows the path of two chains that have mixed (with marginal distribution to the right of panel); B shows two chains that have not mixed.

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 $\hat{R}$ (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 §11.1). 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

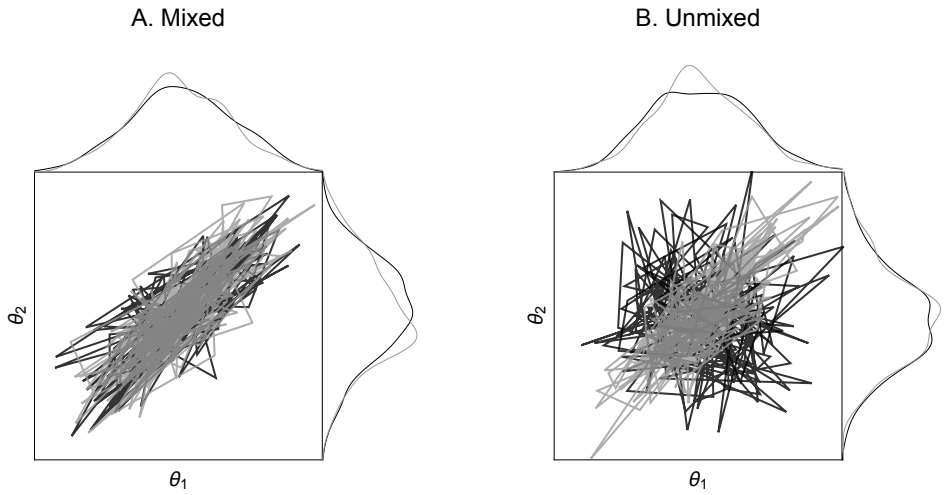

Figure 2: **Chain prediction based on the joint distribution of two parameters where each chain's marginals are the same.** A shows the path of two chains that have mixed resulting in similar sampling distributions (to the right and above each panel); B shows two chains that have not mixed.

use an iterative approach known as "boosting", where each subsequent decision tree aims 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 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 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 §11.2). 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

Algorithm 1 R^* calculation

Given chain-wise draws from the target, $\{X^{\{1\}}, X^{\{2\}}, ..., X^{\{N\}}\}$ and a test set length, S_{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: $\text{ML}(x|X_{\text{train}}) \rightarrow c$
for $s = 1$ to S_{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^*

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 over which to search for an optimal split (Boehmke and Greenwell, 2019, Chapter 11). Our experiments replicate these results (see §11.2), 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: $\boldsymbol{p}^{\{s\}} = (p_1^{\{s\}}, p_2^{\{s\}}, ..., p_N^{\{s\}})$, 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^*.

Algorithm 2 Procedure to generate I draws of R^*

Given test data X_{test}, number of chains N, number of iterations I, and fitted model, $\text{ML}(x|X_{\text{train}}) \rightarrow (p_1, p_2, ..., p_N)$:
for $i = 1$ to I **do**
  for $s = 1$ to S_{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 $(R^{*\{1\}}, R^{*\{2\}}, ..., R^{*\{I\}})$

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. 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) and different draws from the ML model-predicted probability simplex (§3.1); and to differing numbers of chains (§3.1). 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 $t = 1, 2, ..., 2000$. 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.

Example	Relevance	Section
Autoregressive	**Examining R^* and sensitivities to its calculation**	3.1
	Detecting heterogeneous chain variance using R^*	3.1
	Stochasticity in R^*	3.1
	Generating R^* uncertainty measure	3.1
	Sensitivity of R^* to number of chains	3.1
Multivariate normals	**Detecting convergence in joint distributions**	3.2
	Unconverged joint distribution in bivariate normal	3.2
	High correlations between dims in 250D normal	3.2
	Measuring contributions of variables to poor convergence	3.2
Cauchy	**Detecting convergence for long-tailed distributions**	3.3
	Comparing R^* and existing measures to objective convergence	3.3
Eight schools model	**Hierarchical Bayesian model slow convergence**	3.4
Wide multivariate normal	**Detecting convergence when # draws $\sim$ # dims**	7
Non-stationary marginals	**Detecting time-varying sampling distributions**	8
	Trends in mean across all chains	8.1
	Trends in mean in a single dimension	8.2
	Trends in covariance	8.3
	Sensitivity of R^* to chain persistence	8.4
Ovarian and prostate models	**Bayesian models with many parameters and multimodal posteriors**	9
Discrete Markov model	**Evaluating R^* on discrete parameter models**	10
	Small state-space	10.1
	Large state-space	10.2
Various	**Sensitivity of R^* to ML model**	11
	Comparing different ML classifiers	11.1
	Sensitivity of R^* to GBM and RF hyperparameters	11.2
Multivariate normal & Student-t dists.	**Comparing GBM and RF classifiers**	12
	Detecting convergence in joint distributions	12.1
	Tail convergence	12.2

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

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 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 rank-normalised split-$\widehat{R}$ for each replicate; as for R^*, this metric indicates the chains had not converged in all replicates because $\widehat{R} > 1.01$.

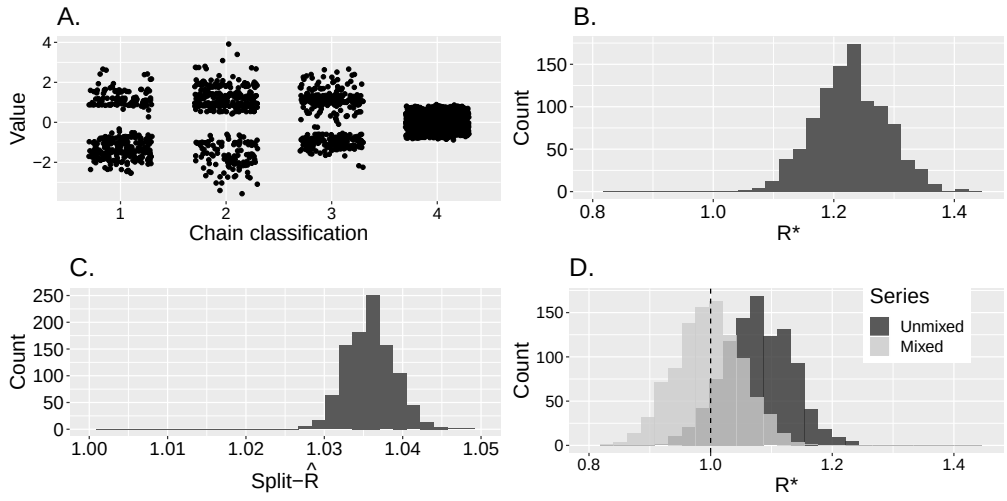

Figure 3: **Autoregressive example.** A shows how the GBM's classifications vary according to the draw's value for an example model fit; B shows R^* values generated by Algorithm 1 across 1000 replicate datasets; C shows corresponding rank-normalised split-$\widehat{R}$ values for each of the 1000 replicates; 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" 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 §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 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 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, we consider an alternative approach based on bootstrapping a single ML model's predictions.

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.

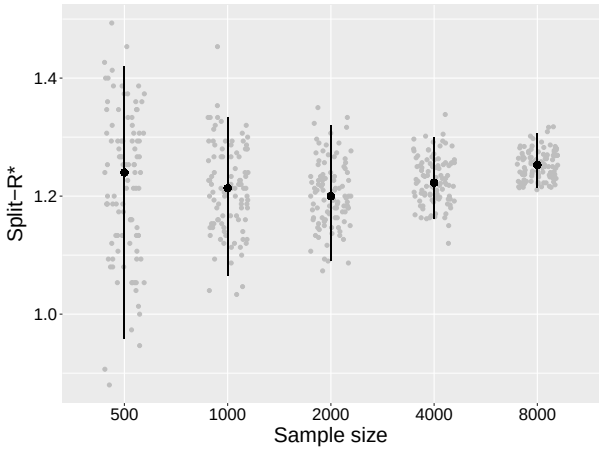

Figure 4: **Autoregressive example: R^* stochasticity.** The horizontal axis shows sample size; the vertical axis shows the value of R^* calculated as per Algorithm 1 applied to chains split into two halves. Grey points show the value of R^* for each replicate (jitter was added to point positions). Black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

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. 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.

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 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 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.

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.

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

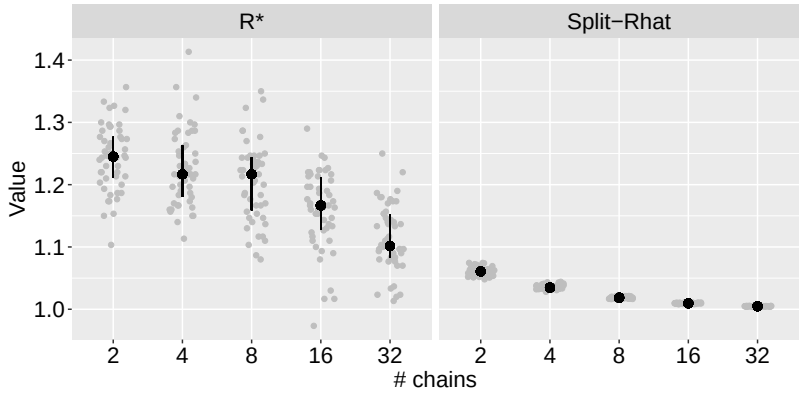

Figure 5: **Autoregressive example: sensitivity to number of chains.** The horizontal axis shows the number of chains used in the data generating process described in §3.1. The vertical axis shows the value of R^* as calculated by Algorithm 1 (left panel) on chains split into two halves, and rank-normalised 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.

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 Fig. 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 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.

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,

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

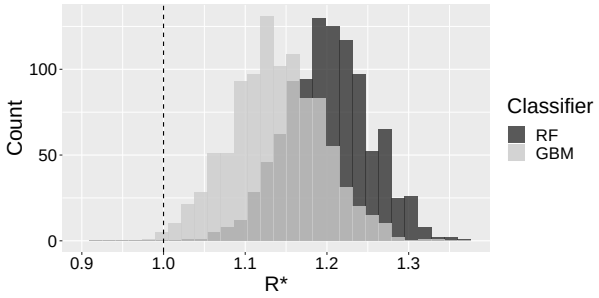

Figure 6: **Bivariate normal example.** The distribution for R^* across 1000 draws as calculated using Algorithm 2 for both the GBM and RF classifiers.

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 Fig. 7A (the equivalent plot for a RF classifier is similar and shown in Fig. 12). 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 (Fig. 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,

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

where $\boldsymbol{L}$ is the Cholesky decomposition of the covariance matrix, $\boldsymbol{A}^{-1}$. Fig. 7A shows the R^* distribution resultant from 10,000 NUTS iterations in this case: now the distribution has mean $R^* = 1.00$. Fig. 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.

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 Fig. 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 Fig. 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.

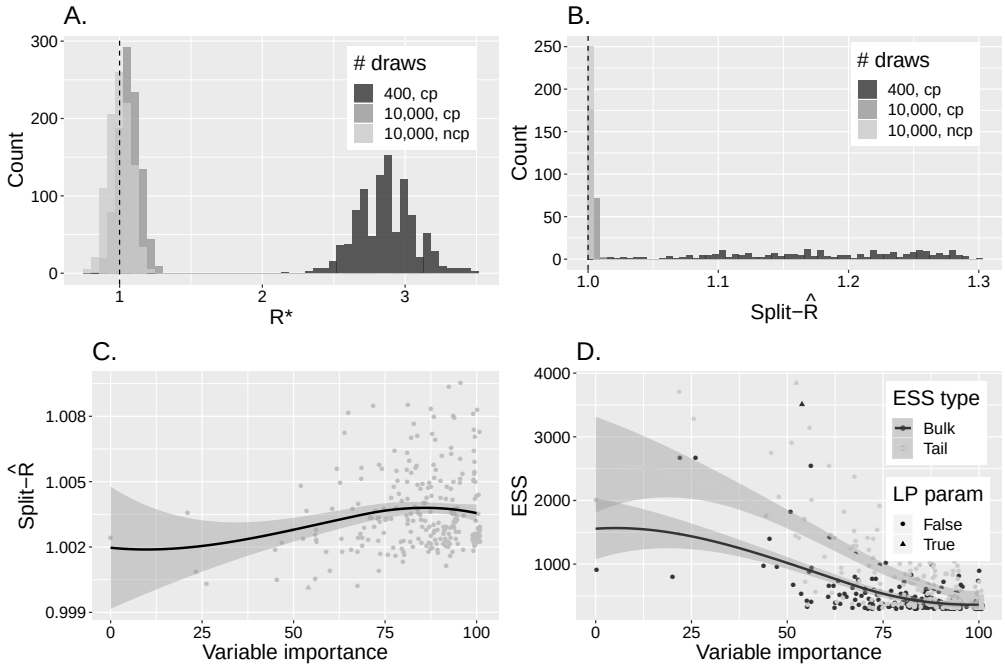

Figure 7: **Multivariate normal example with 250 dimensions.** A shows R^* distributions obtained for two MCMC samples (of differing numbers of draws: 400 and 10,000) from the centered parameterisation ("cp") and one from the non-centered version ("ncp"; with 10,000 draws); B shows the rank-normalised 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.

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,

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

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 Fig. 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, 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. 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 $\widehat{R} > 1.01$, 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. 8, we show the results for these longer runs. In these, the alternative parameterisation now has 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.

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.

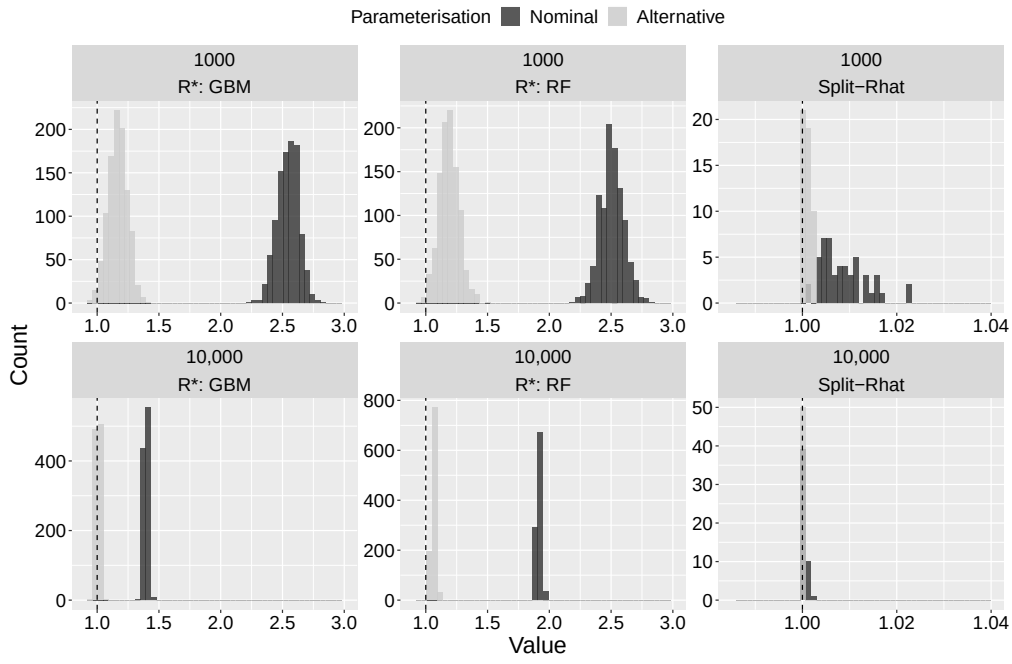

Figure 8: **Cauchy example.** Rows show convergence results for MCMC runs with 1000 (top) and 10,000 (bottom; obtained by thinning iterations by a factor of 3) post-warm-up iterations (each with half iterations discarded as warm-up) for each of 4 chains. Columns show the R^* distributions from GBM (left)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

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. 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 Figs. 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 Fig. 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; Fig. 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 $\widehat{R} > 1.07$ indicating more draws are needed. In Fig. 9D, we plot R^* against iteration for both models 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 Figs. 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 Fig. 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^* (Fig. 9D) appears at least, qualitatively, most similar to *quantile-*R^2.

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,

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

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,

$$\tilde{\theta}_j \sim \text{normal}(0, 1),$$

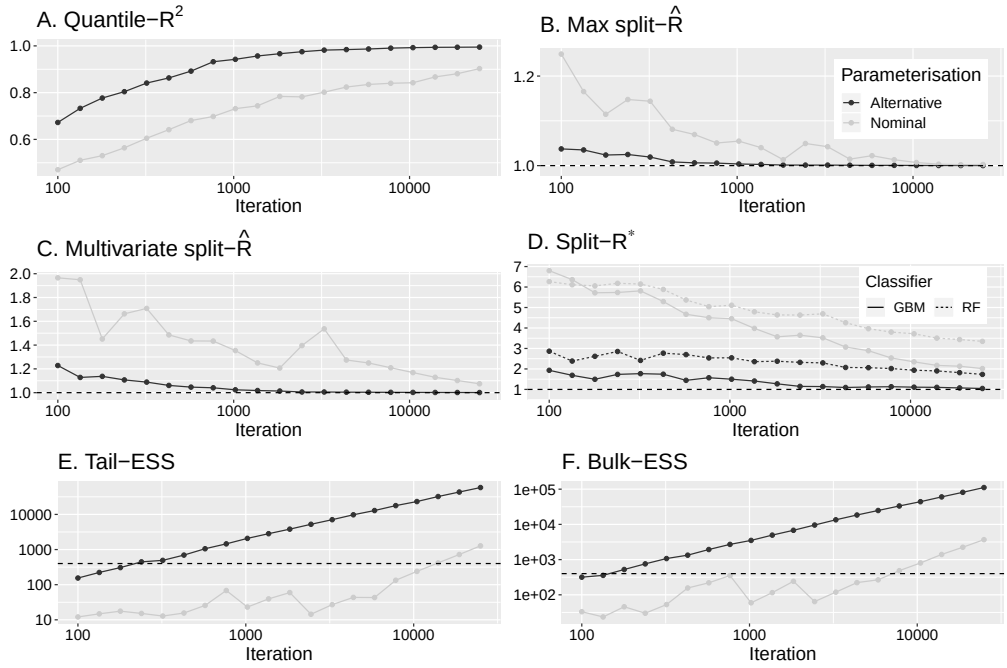

Figure 9: **Measuring convergence for the Cauchy model.** A shows a measure of convergence, the mean quantile R^2, that requires knowing the target distribution; B shows the maximum value of split-$\widehat{R}$ across each of the 50 dimensions of the target; C shows the multivariate split-$\widehat{R}$ 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. Dashed lines indicate recommended thresholds for each convergence statistic.

$$\theta_j = \mu + \tau\tilde{\theta}_j,$$
$$y_j \sim \text{normal}(\theta_j, \sigma_j).$$

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 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 Fig.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 Fig.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. 13 shows the equivalent of Fig. 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 Fig. 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.

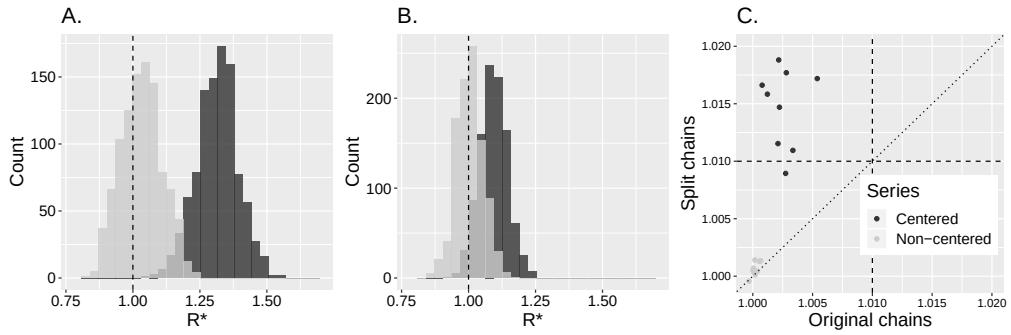

Figure 10: **Eight schools example: R^* distributions.** A shows draws from the R^* distribution when splitting chains in two (resulting in 8 chains); B shows the same but using the 4 original chains; C shows rank-normalised $\widehat{R}$ for original 4 chains versus those for the 8 chains case for all 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.

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 Fig. 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.

Fig. 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.

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 §7, 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 §8, 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

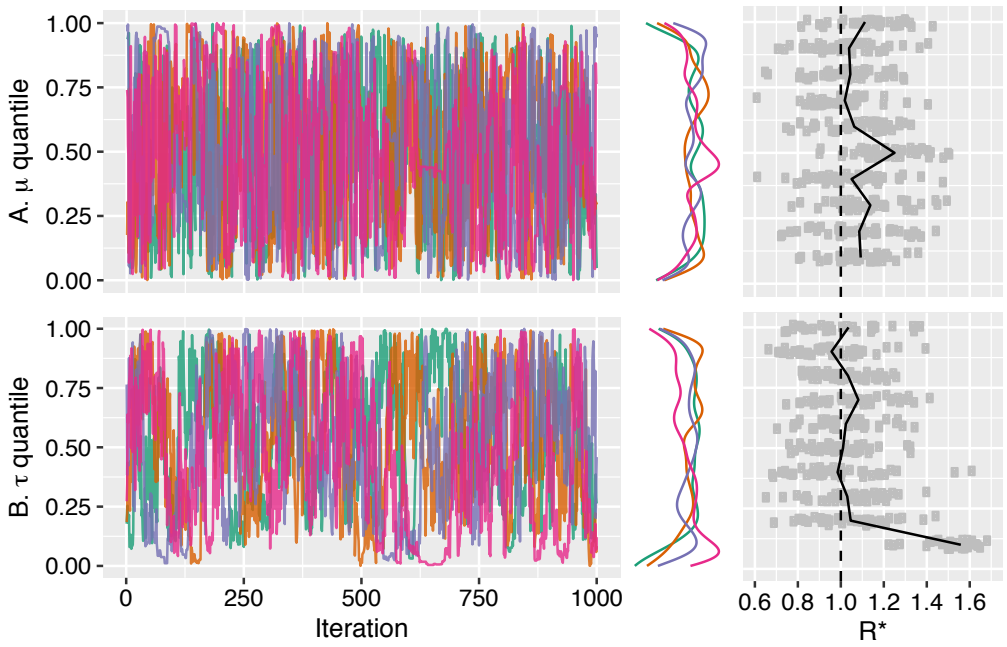

Figure 11: **Eight schools example: quantile R^* plots.** Row A shows plots for μ; Row B for τ. In each row, we show the path of the four individual chains above and 40 R^* draws obtained using Algorithm 2 for each parameter quantile below. To the right of each trace plot, we show the marginal distribution 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.

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 §9, 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 §10, we evaluate R^* on univariate discrete examples: one with four states (in §10.1); another, with a larger state-space consisting of 20 states (in §10.2). 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 §11, we investigate how two decisions about classifiers which classifier to use (in §11.1) and what hyperparameters to use for it (in §11.2) affect calculation of R^*. In §11.1, 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 §11.2, 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 §12, we use a slew of examples to compare R^* calculated using GBM and RF classifiers using our suggested default hyperparameter sets (given in §2). In §12.1, we compare how both methods are able to diagnose lack of convergence in a joint distribution (using a multivariate normal target). In §12.2, 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 §12 for more information). Collectively, the results of §12.1 and §12.2 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 §2 and §11.2): 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 (§11.2), and useful heuristics for adapting these with target dimensions are known, which we follow here, as described in §2. 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 §11.1), 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 §9, 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 $\mathcal{O}(m_{\text{try}} n_{\text{tree}} N \log N)$

(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, §3.3 and §9, 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 $1, 2, ..., K$ 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

5 Diagnosing convergence in joint distributions: multivariate normal models supplementary

In Fig. 12, we compare the performance of GBM and RF classifiers on the multivariate normal example described in §3.2. The two classifiers produced similar R^* distributions across the various examples: for the 400 draw example, the GBM R^* distribution had a mean of 2.86 and the RF equivalent was 3.06; for the example with 10,000 samples from the centered parameterisation, the GBM classifier had a R^* distribution mean of 1.07 versus 1.08 from the RF; for the example with 10,000 samples from the non-centered parameterisation, both classifiers produced a mean of 1.00.

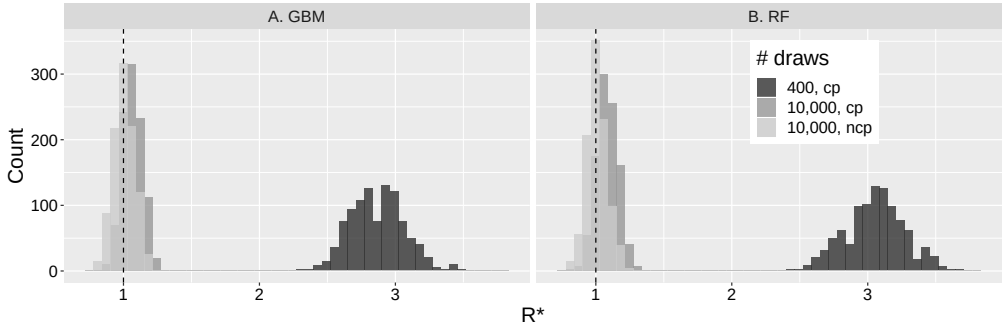

Figure 12: **Multivariate normal example: GBM (A) versus RF (B) classifiers.** Shows R^* distributions obtained for two MCMC samples (of differing numbers of draws: 400 and 10,000) from the centered parameterisation ("cp") and one from the non-centered version ("ncp"; with 10,000 draws). Here, we show 1000 R^* draws by Algorithm 2 for each MCMC run.

6 Hierarchical model: Eight schools model supplementary

Fig. 13 shows the equivalent of Fig. 10 except using a RF classifier.

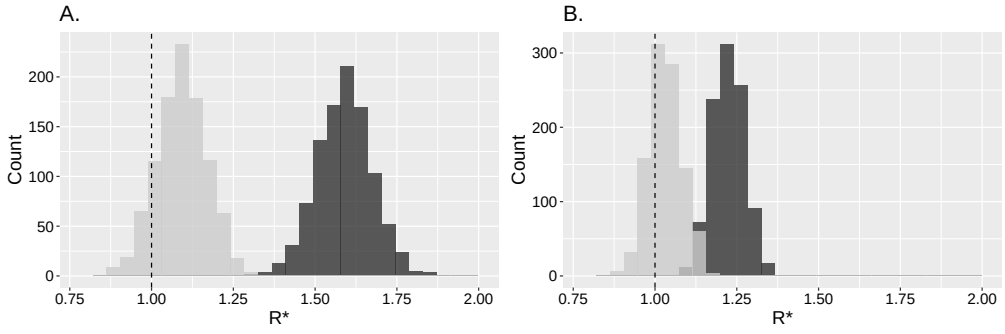

Figure 13: **Eight schools example: R^* distributions for a RF classifier.** A shows draws from the R^* distribution when splitting chains in two (resulting in 8 chains); B shows the same but using the 4 original chains. The MCMC samples comprised 2000 draws in all cases, with 1000 used as post-warm-up iterations. In panels A and B, the plots show 1000 R^* draws using Algorithm 2 for each parameterisation.

7 Wide datasets: multivariate normal

As the number of parameter dimensions increases, it might be thought that ML algorithms will overfit the data, and, hence, testing set classification would be poor; leading to unreliable determinations of convergence. To test this hypothesis, we investigated two scenarios using a multivariate normal target.

7.1 250-dimensional model

In the first of these, we used the 250-dimensional multivariate normal of eq. (3.2) with 250 post-warm-up iterations (after 250 warm-up iterations) for each of 4 chains from Stan's NUTS to calculate R^* distributions as in Algorithm 2. Here, we considered both the centered and non-centered parameterisations, where, in both cases, the number of iterations is comparable to the number of parameters, so the training data is relatively "wide". We also calculated R^* distributions using both the GBM and RF classifiers: the R^* distribution in each case is shown in Figs. 14A&B. This figure shows that, across both parameterisations, convergence has not yet been reached since all R^* distributions are shifted rightwards of 1. The same conclusion is reached if rank-normalised split-$\widehat{R}$ is used instead (Fig. 15A), since, for both parameterisations, some of the parameters had $\widehat{R} > 1.01$. Using bulk- or tail-ESS instead, we conclude that the non-centered parameterisation shows signs of convergence whereas the centered does not (Fig. 15B&C).

7.2 10,000-dimensional model

We next consider a more challenging example – a target distribution with 10,000 dimensions. In this case, we assume independent standard normals for each dimension. In Figs. 14C&D, we plot the R^* distributions from GBM and RF classifiers for two MCMC runs targeting this distribution: one with 400 iterations, the other with 1000. In all cases, the distributions were right of $R^* = 1$, indicating non-convergence. These results were also echoed by rank-normalised split-$\widehat{R}$, with 19% of dimensions having $\widehat{R} > 1.01$ for the 400 iteration case and 3% for the 1000 iteration case.

Overall, the examples in this section suggest that R^* is a conservative measure of convergence: when there are not enough draws, it will tend to diagnose non-convergence. We also note that the statistic took comparable time to calculate relative to existing convergence diagnostics on a desktop computer.

8 Non-stationary marginals

If a Markov chain does not mix with itself, this also indicates that convergence has not occurred (Gelman et al., 2013). In this section, we investigate whether R^* can detect non-stationary sampling distributions. In all the examples, we present results for both GBM and RF classifiers.

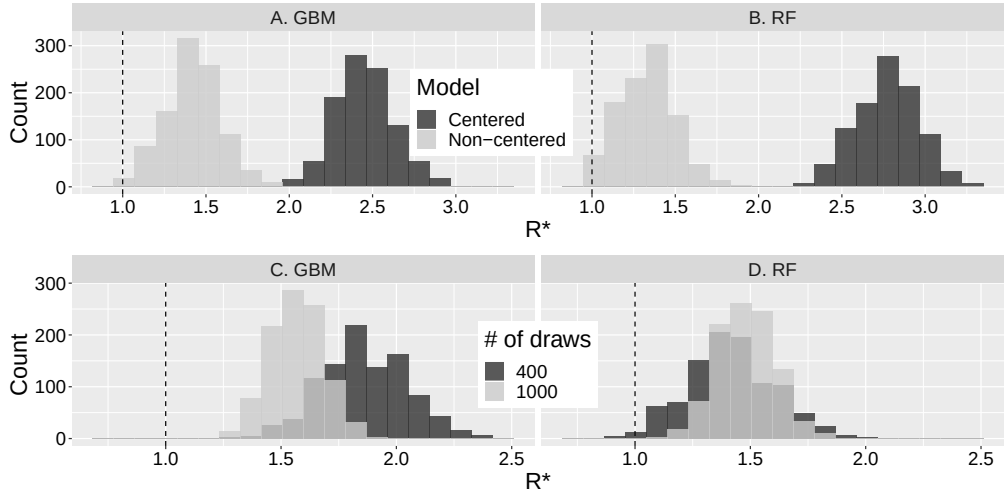

Figure 14: **Wide data examples.** Top row shows the R^* distribution for the 250-dimensional example in §7 with 250 post-warm-up iterations per chain from Stan's NUTS algorithm across both model parameterisations; bottom row shows R^* distribution for the 10,000-dimensional example with 400 and 1000 MCMC iterations per chain (although the first half of these were discarded as warm-up). In the left column, results for the GBM classifier are shown; in the right, we show the same for the RF classifier. In all cases, 1000 draws of R^* are plotted as generated by Algorithm 2 for a single MCMC run composed of 4 chains.

8.1 Trending mean across all chains

We first recapitulate an example from appendix A in Vehtari et al. (2020). This example showed that split-$\widehat{R}$ could detect non-convergence caused by shifts in sampling distributions over time: in their case, they analysed chains with common linear trends in mean. Specifically, they first generated 4 chains by random sampling from a univariate normal distribution, then added a common time trend to each chain, resulting in a univariate distribution whose mean increased during sampling. We first repeat this analysis but using R^* rather than $\widehat{R}$: in Fig. 16, we show these results. In column A, we show the results for R^* calculated on the 4 chains that ran; column B shows the same calculation but after the chains are split into two equal halves. The rows show the range of sample sizes investigated: 250, 1000 and 4000; the horizontal axis shows the magnitude of time trend added to each sample; for all parameter sets, we run 10 replicates. The points and triangles show R^* derived from GBM and RF classifiers, respectively. This plot mirrors Fig. 4 in the supplementary materials of Vehtari et al. (2020) and shows that, without splitting the chains, R^* does not increase with trend whereas, after splitting, it does. As expected, split-R^* is more reliably able to detect non-convergence as sample size increases.

These results make intuitive sense: without systematic between-chain variation, it is

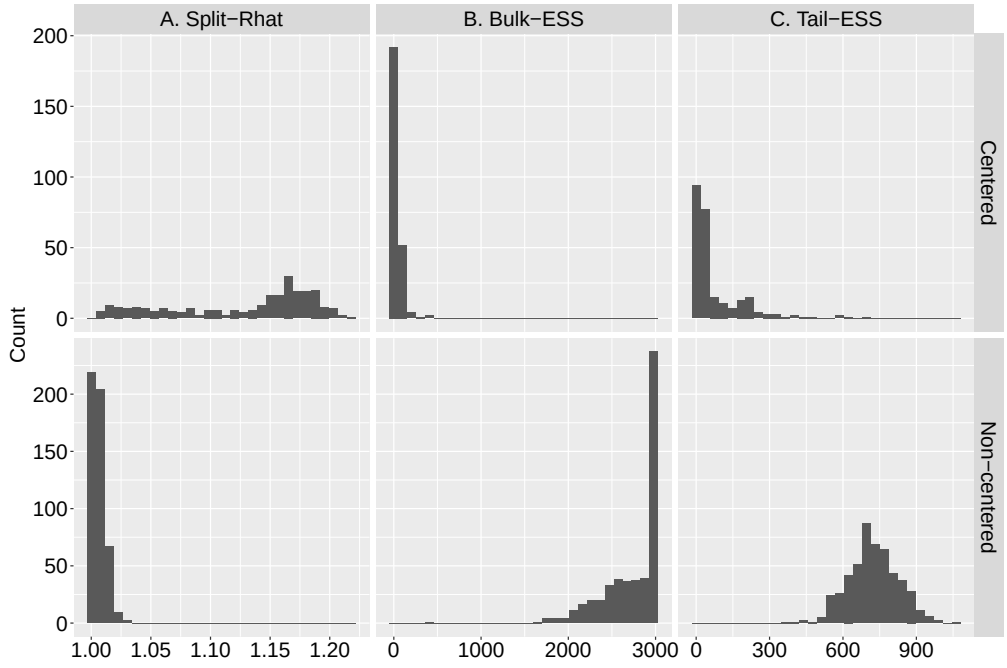

Figure 15: **Wide data 250-dimensional example: established diagnostics.** The top row shows the results for the centered parameterisation; the bottom row for the non-centered. Column A shows split-$\widehat{R}$; columns B and C show the bulk- and tail-ESS; in each case the statistics are displayed for all model parameters and were calculated using 250 post-warm-up draws from Stan's NUTS algorithm. Note, that it is possible for the ESS to exceed the actual sample size if there is negative autocorrelation in the Markov chains' values. In both cases, the results correspond to a single MCMC run composed of 4 chains.

not possible to reliably determine which of them caused a particular observation. In this case, because all chains exhibited the same secular trends over time, there would not be differences in their marginals. By splitting chains into two – the first half being the early phase, and the second half being the later phase with higher mean – this forces differences in the marginals. This meant it was possible to reliably pick whether an observation was caused by an early phase chain or a later one. As such, we recommend that R^* always be calculated using split chains as is recommended for $\widehat{R}$ (Carpenter et al., 2017; Vehtari et al., 2020).

8.2 Trending mean in a single dimension

We next consider whether split-R^* can detect non-convergence when only a single dimension trends. In Fig. 17, we show how R^* performs across a range of target dimensions.

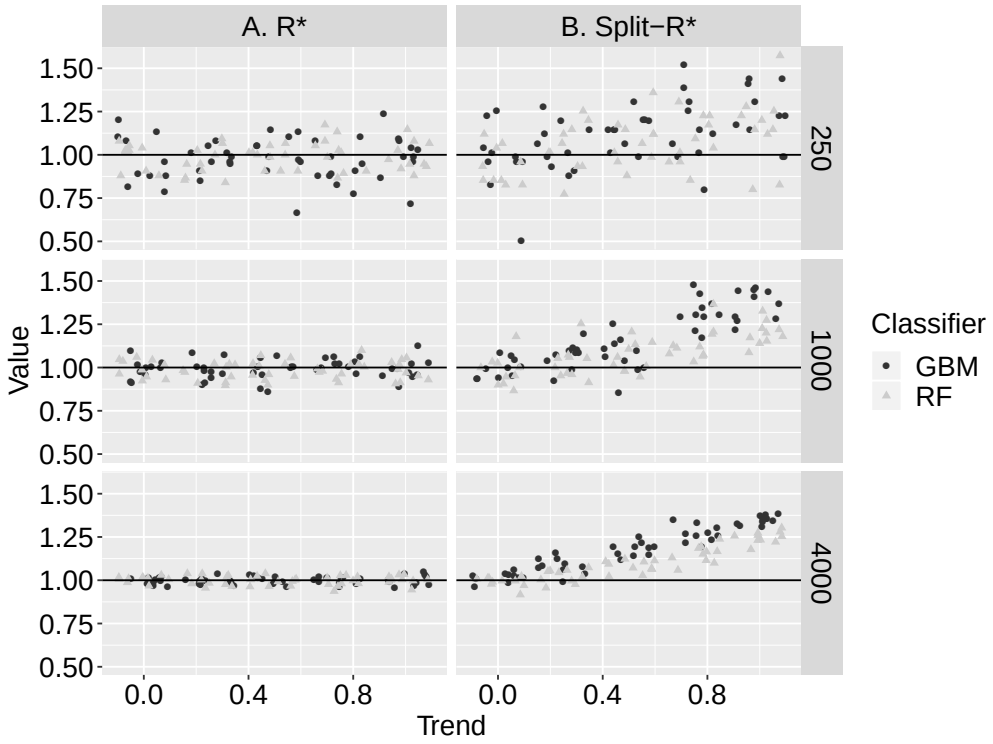

Figure 16: **Univariate trends example.** Points and triangles represent results for the GBM and RF classifiers, respectively. Column A represents results for R^* calculated using Algorithm 1 on the 4 chains; column B shows the same calculation after each chain is split into two halves. The rows present the differing sample sizes. The horizontal axis measures (half) the change in mean across the whole sample: so a value "1" indicates the mean increases by 2 units from the start to end of sampling. At each parameter set, 10 replicates were run and jitter was added to the points.

In the simulations here, all dimensions bar one are stationary; the remaining dimension has a linear trend added to it. In all cases, across both GBM and RF classifiers, split-R^* increased with trend. Indeed, differences in the typical values of this metric were not apparent across the different target dimensions considered. This suggests split-R^* can robustly determine chain identity if only a single dimension has a non-stationary sampling distribution.

8.3 Trending covariance

Means that trend over time is one form of non-stationarity; another is a time-varying covariance. Next, we consider a bivariate normal with (constant) standard normal marginals but where the correlation between dimensions trends over time. Specifically,

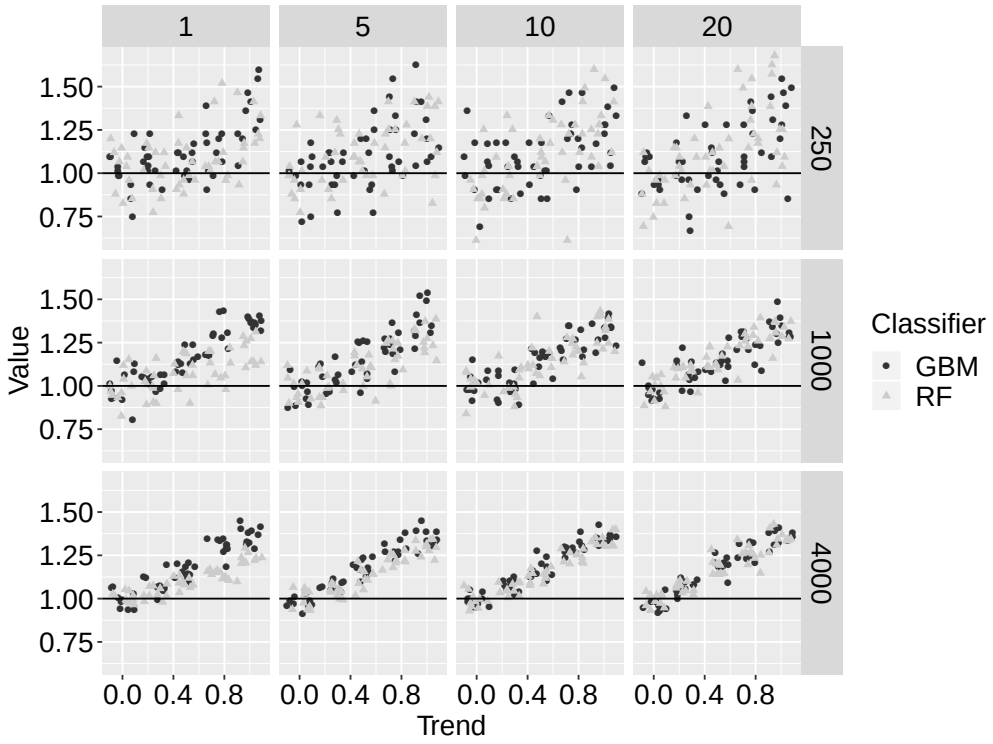

Figure 17: **Multivariate trends example with split-R^*.** Points and triangles represent results for the GBM and RF classifiers, respectively. The columns present different dimensionalities of the target distribution; the rows present different sample sizes. The horizontal axis measures (half) the change in mean across the single dimension that had a trend added to it: a value "1" indicates its mean increases by 2 units from the start to end of sampling; all other dimensions (if dimensions exceeded 1) had stationary distributions. At each parameter set, 10 replicates were run and jitter was added to the points.

we allow the correlation to increase linearly from $-\rho$ and ρ throughout the course of simulations and use i.i.d. draws from the process across 4 "chains". Again, as before, R^* calculated on unsplit chains is unable to detect this form of non-stationarity, since there are no inter-chain differences in the sampling distribution. Similarly, split-$\widehat{R}$ does not detect this form of non-convergence since the marginal distribution across chains does not vary over time (Fig. 18A). By contrast, split-R^* can (Fig. 18B), since it uses all information in the samples, including the covariance structure.

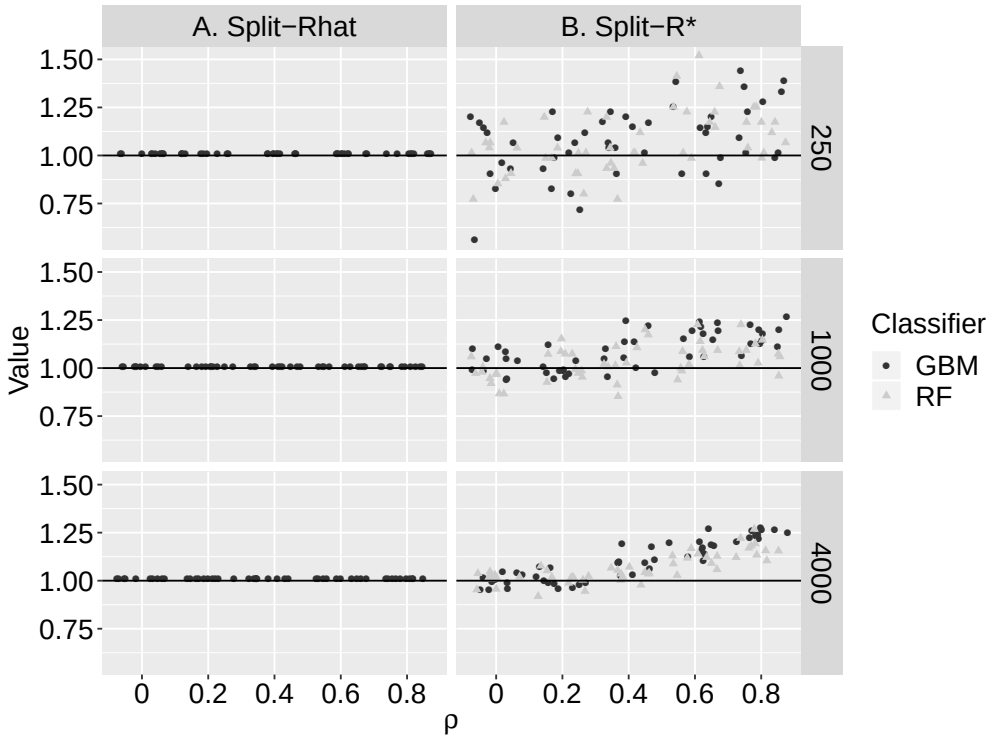

Figure 18: **Bivariate normal with trending correlation example.** Points and triangles represent results for the GBM and RF classifiers, respectively. Column A shows the results for split-$\widehat{R}$; column B for R^*; the rows present the differing sample sizes. The horizontal axis measures (half) the change in correlation across the whole sample: so a value "0.5" indicates the correlation increases by 1 unit (from -0.5 to 0.5) from the start to end of sampling. At each parameter set, 10 replicates were run and jitter was added to the points.

8.4 Chain persistence

When forming training and testing sets as part of the ML algorithm used to determine R^*, the testing set is effectively treated as a sort of "independent" hold-out dataset. Markov chains, in general, have persistence, meaning that the test set will not be truly independent and can – according to the level of autocorrelation in the chains – be highly related to the training set. In this section, we investigate how this autocorrelation affects the performance of R^*. The difficulty with this question is that higher chain autocorrelation typically means the sampling distribution is a rougher approximation of the target, so R^* should be higher due to the properties of the sampling distribution. It could also be higher because the training set is less distinct from the testing set.

To investigate this, we generated AR(1) processes (as defined in eq. (3.1)) with

autocorrelations, ρ, ranging from 0.8-1 and, in each case, calculated R^* via Algorithm 1. Note, that only when $|\rho| < 1$ is the marginal distribution defined by this process itself stationary; at $\rho = 1$, its variance increases linearly with time, so, by definition, is not converged. In Fig. 19, we show the results of these simulations across various numbers of iterations: 250, 1000 and 4000 (different panels). In all cases, $R^* > 1$ whenever $\rho = 1$, indicating lack of convergence. As ρ declined, so did R^*.

Notably, as the number of iterations increased, the values of R^* for $\rho < 1$ declined, whereas R^* for $\rho = 1$ actually increased: this is most easily seen by examining Fig. 20 (which shows the same data as in Fig. 19 but in a different way). This characteristic is exactly as desired: larger sample sizes should yield a sampling distribution closer to convergence for $\rho < 1$; the same for $\rho = 1$ produces distributions that are no closer to convergence, and more samples allows better determination of this non-convergence.

Overall, it seems that R^* is a conservative measure and with greater chain autocorrelation it suggests more draws are necessary for convergence.

9 Many parameter models: ovarian and prostate analysis

In this section, we analyse two Bayesian models both fit to real data. In both examples, we illustrate R^* distributions derived only from GBM classifiers. The first is a logistic regression model fit to microarray ovarian cancer data with 54 data points and 1536 predictor variables; overall, this model has 4719 parameters. Since there are relatively few data points relative to the number of predictors, regularised horseshoe priors are specified on the regression coefficients (Piironen and Vehtari, 2017) since most are expected to be zero. This dataset has been used for benchmarking in the past (see, for example, Schummer et al. (1999); Hernández-Lobato et al. (2010); Paananen et al. (2019)) and is known to result in a multimodal posterior.

The other dataset we use is of similar form but for prostate cancer. The original form of the dataset is described here: Singh et al. (2002). We use a filtered version of the dataset as detailed here: Yang et al. (2006), which we also analysed using logistic regression with regularised horseshoe priors. It has 18,105 parameters in total.

For each model, we consider two MCMC runs: one with 4 chains run with 800 thinned post-warm-up iterations (9000 total iterations with 1000 discarded as warm-up; 8000 post-warm-up iterations thinned by a factor of 10); another with 16 chains run with 1000 post-warm-up iterations each (500 warm-up iterations discarded and no thinning).

For the ovarian model, we show the results in Fig. 21. In Fig. 21A, we show the R^* distributions for each model run, which show that, whereas the "long" model run has converged, the "short" one has yet to do so. Whilst it is harder to discern, this pattern is mirrored in Fig. 21B, since the long model has $\widehat{R} < 1.01$ for all parameters, whereas the short model had 83 parameters where this was not the case. Similarly, in Figs. 21C and 21D, which show the bulk-ESS and tail-ESS respectively, it is evident that, for the short model, there remain a few parameters with low effective sample sizes, whereas the long model has more consistent values for this metric.

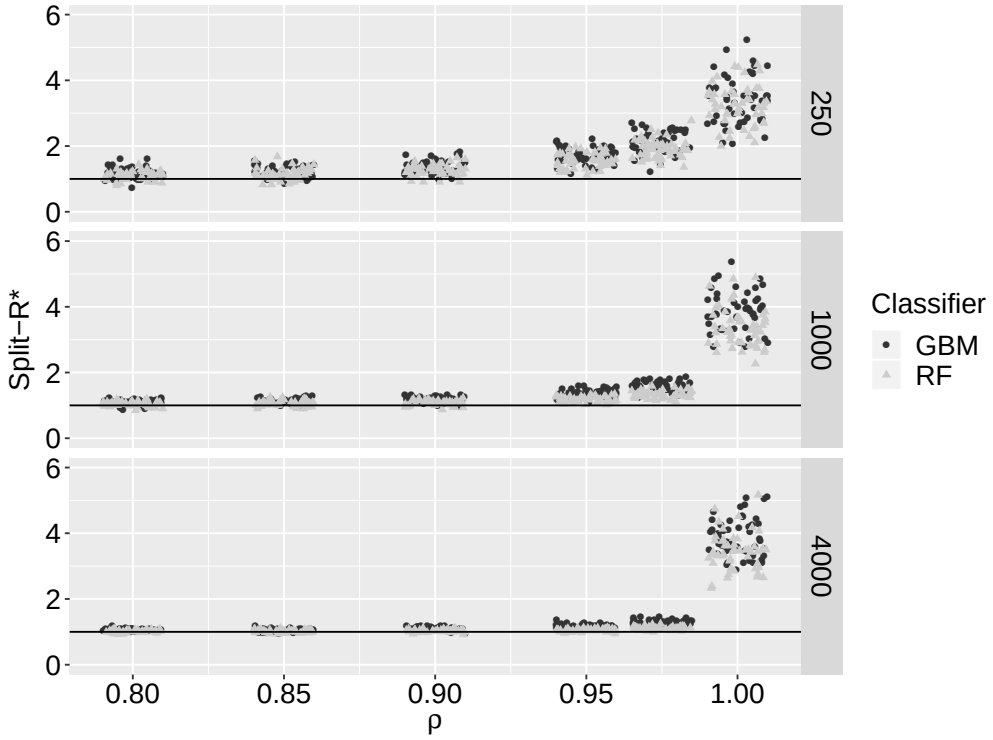

Figure 19: **Non-stationary distribution: AR(1) example.** Points and triangles represent results for the GBM and RF classifiers, respectively. The horizontal axis measures the autocorrelation of the AR(1) processes; the vertical axis shows the value of R^* calculated on chains split in half; each panel shows a different number of iterations. At each parameter set, 10 replicates were run and jitter was added to the points. The black line shows $R^* = 1$.

For the prostate model, we show the results in Fig. 22. Since this model has nearly four times as many parameters as the ovarian model, it was more computationally expensive to estimate R^* for it. To handle this, we thinned down the parameters by a factor of 5 for the long model, recognising that, of course, this measure will make it more likely that we diagnose convergence. Despite this, both R^* measures indicated that the MCMC runs had yet to converge (Fig. 22A), which was mirrored by the other metrics considered (Figs. 22B,C&D).

10 Discrete distributions

In this section, we compare the performance of R^* and $\widehat{R}$ on a univariate discrete target distribution. To do so, we generate draws from four discrete Markov chains, following a

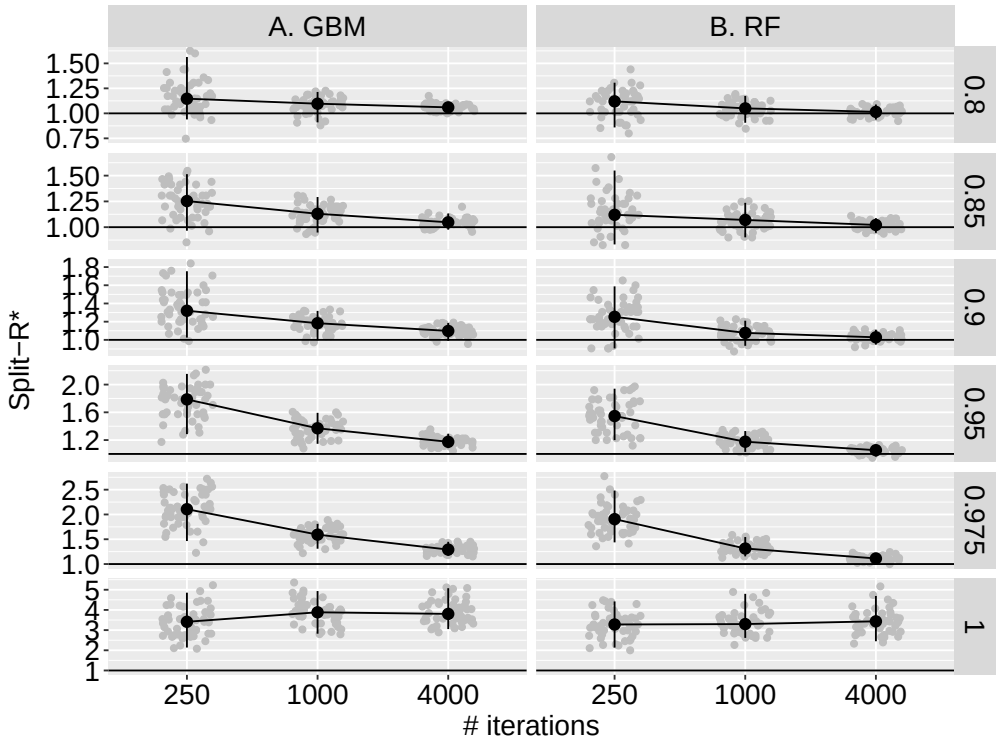

Figure 20: **Non-stationary distribution: AR(1) example alternative view.** Column A shows the results for a GBM classifier; column B for a RF classifier. The horizontal axis measures the number of iterations; the vertical axis shows the value of R^* calculated on chains split in half; each panel shows a different value of autocorrelation. At each parameter set, 10 replicates were run and jitter was added to the points. The black dots show the median R^* values at each parameter set; upper and lower whiskers show 2.5% and 97.5% quantiles. The black horizontal line shows $R^* = 1$.

1st order Markov process with transition probabilities,

$$p_{ij} = \Pr(X_{n+1} = j | X_n = i), \tag{10.1}$$

where p_{ij} is the probability of transitioning from state i at time n to state j at time $n + 1$, and $X(0) = 1$. The target distribution is, hence, the stationary distribution of this process.

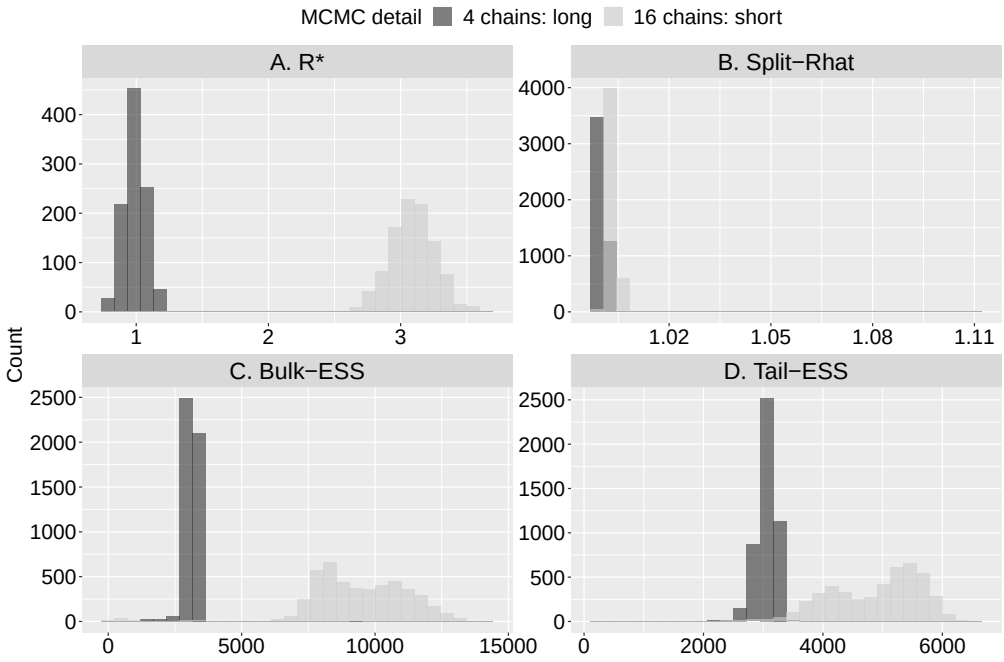

Figure 21: **Ovarian example.** In all plots, we show the results from two model runs: one with 4 chains run with 800 thinned post-warm-up iterations (9000 total iterations with 1000 discarded as warm-up; 8000 post-warm-up iterations thinned by a factor of 10); another with 16 chains run with 1000 post-warm-up iterations each (500 warm-up iterations discarded and no thinning). In A, R^* distributions (with 1000 draws each using Algorithm 2) using the GBM classifier are shown; B shows rank-normalised split-$\widehat{R}$ values across all parameters; C shows bulk-ESS across all parameters; and D shows tail-ESS across parameters.

10.1 Small state-space

We first consider a univariate discrete distribution with four states. In three of the four chains, we generate draws using a transition probability matrix:

$$\boldsymbol{P} = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/3 & 1/6 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{10.2}$$

which implies a stationary target distribution with the following probability simplex $\pi_1 = (11/46, 15/46, 14/46, 6/46)$.

For the fourth chain, we specify differing transition probability matrices in each of three sets of results. In the first, we assume $\boldsymbol{P}_1 = \boldsymbol{P}$ (i.e. the same as used for the other

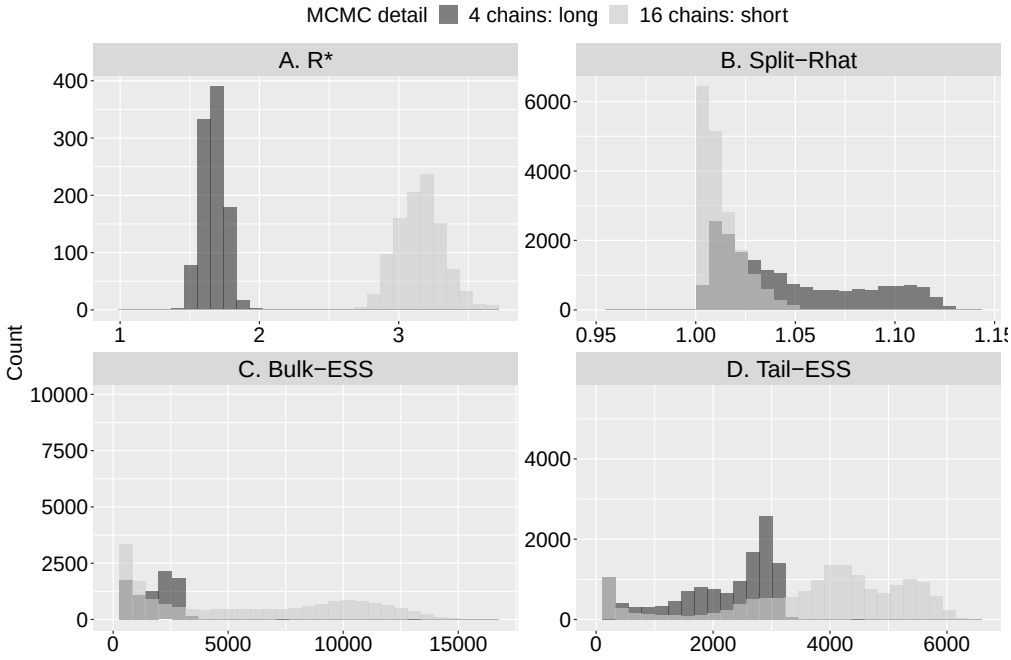

Figure 22: **Prostate example.** In all plots, we show the results from two model runs: one with 4 chains run with 800 thinned post-warm-up iterations (9000 total iterations with 1000 discarded as warm-up; 8000 post-warm-up iterations thinned by a factor of 10); another with 16 chains run with 1000 post-warm-up iterations each (500 warm-up iterations discarded and no thinning). In A, R^* distributions (with 1000 draws each using Algorithm 2) using the GBM classifier are shown – for the long run, these were calculated after thinning the parameters by a factor of 5; B shows rank-normalised split-$\widehat{R}$ values across all parameters; C shows bulk-ESS across all parameters; and D shows tail-ESS across parameters.

three chains); in the second and third cases, we assume this transition matrix is given by:

$$\boldsymbol{P}_2 = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/3 & 1/6 \\ 5/8 & 1/8 & 1/8 & 1/8 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{P}_3 = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1/3 & 1/6 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{10.3}$$

These correspond to stationary probability simplices:

$$\pi_2 = (71/198, 17/66, 10/33, 8/99), \quad \pi_3 = (4/9, 2/9, 8/27, 1/27). \tag{10.4}$$

The three stationary distributions corresponding to each transition probability matrix are shown in Fig. 23. This illustrates that there is an ordering among the stationary

distributions $\pi_1 \sim \pi_2 \sim \pi_3$, meaning π_1 is most similar to π_2, and π_3 is most similar to π_2. Put another way, π_3 is most different from π_1.

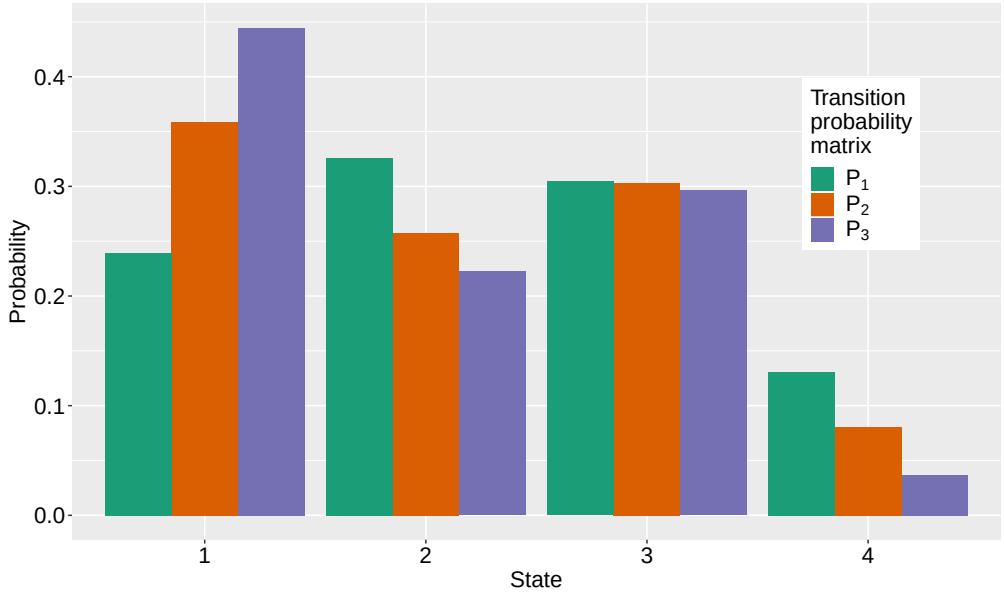

Figure 23: **Discrete distribution example: stationary distributions.** The three colours illustrate the stationary distributions of the Markov chains corresponding to the matrices P_1, P_2 and P_3 defined in §10.

For each of these three cases, we consider a range of sample sizes across each of the four chains: 1000, 2000, 5000 and 10,000. At each combination of transition probability matrix (for the fourth chain) and sample size, we perform 40 replicates. In each replicate, we calculate both R^* using a GBM classifier and rank-normalised split-$\widehat{R}$. The results of these experiments are shown in Fig. 24. In the horizontal axes of both panels, we show the sample size. The vertical axis indicates the value of R^* and split-$\widehat{R}$ in panels A and B, respectively. The stacked sub-panels within A and B indicate the transition matrix used for the fourth chain. Since $\boldsymbol{P}_1 = \boldsymbol{P}$, the stationary distribution for the other three chains, both diagnostics should tend towards indicating convergence in these cases, and, for both statistics, this was typically the case. For R^*, the replicates were centred on $R^* = 1$, although the stochasticity of replicates meant that, in some cases, $R^* > 1$. Whereas, $\widehat{R}$ was similarly close to 1 in all cases. Considering the simulations using $\boldsymbol{P}_2$ for the fourth chain transition probability matrix, $R^* > 1$ across a majority of replicates, and this majority increased with sample size. In this case, $1.01 > \widehat{R} > 1$ for all replicates across the various sample sizes. For simulations using $\boldsymbol{P}_3$ for the fourth chain transition probability matrix, $R^* > 1$ across all replicates and the magnitude of median R^* increased with sample size; in this case, $\widehat{R} > 1.01$ in all replicates.

This example shows that R^* can detect poor convergence for discrete parameter

spaces, like $\widehat{R}$.

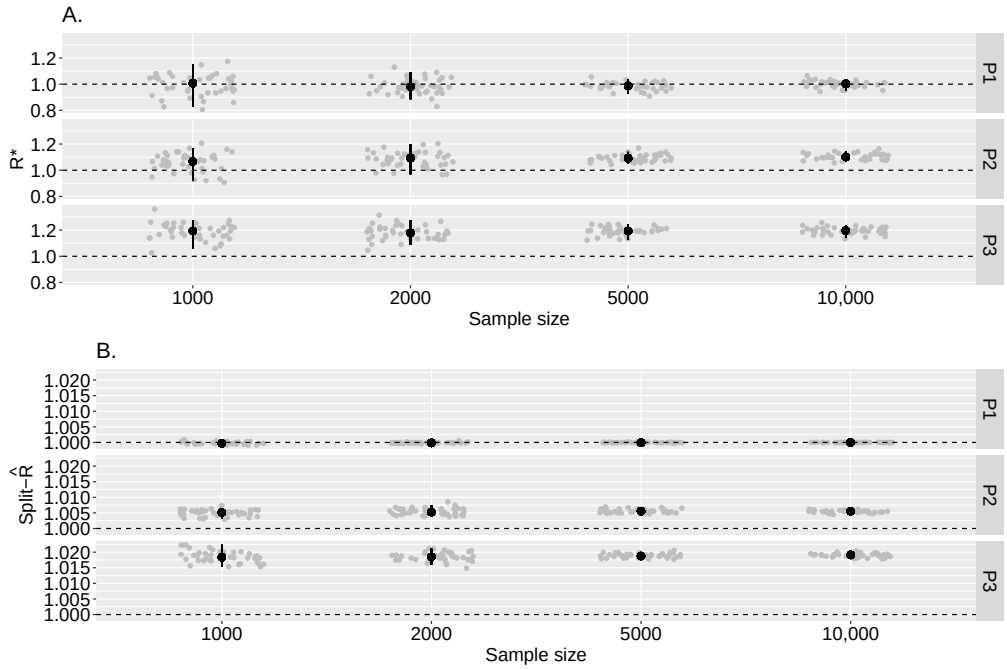

Figure 24: **Discrete distribution example: convergence diagnostics.** On the horizontal axis, we show the sample size of each set of simulations. The panels within A. and B. indicate the transition probability matrix used for the fourth chain (the other three chains always used $\boldsymbol{P} = \boldsymbol{P}_1$). In panel A, grey points indicate R^* from each replicate calculated using Algorithm 1) , and the black point-ranges indicate 2.5% , 50% and 97.5% quantiles across the 40 replicates. In panel B, we show the same, but for split-$\widehat{R}$. The dashed horizontal lines at 1 indicate the convergence threshold in both cases.

10.2 Large state-space

We next consider a larger discrete parameter space: a univariate distribution with 20 states. As in §10.1, we generate draws from three discrete Markov chains that share a common transition probability matrix, $\boldsymbol{P}$. For the fourth (and final) chain, we consider two cases: one where it shares the same matrix, $\boldsymbol{P}_1 = \boldsymbol{P}$; the other, when its Markov process has a different matrix, $\boldsymbol{P}_2$. Rather than write down a given $\boldsymbol{P}$ and $\boldsymbol{P}_2$, we generated these stochastically: for each of the rows, we randomly sampled from from a Dirichlet process with shape vectors consisting of ones (of length 20).

To illustrate the differences between the two transition probability matrices, $\boldsymbol{P}$ and $\boldsymbol{P}_2$, we simulate 100,000 draws from each Markov process. In Fig. 25, we show the marginal distribution obtained across all four chains: where the first three used $\boldsymbol{P}$, and

the last used $\boldsymbol{P}_2$. In this plot, we show only the first four states of the 20-state model. The differences in marginal distributions between the first three chains and the last indicate that there are substantial differences in the target distributions.

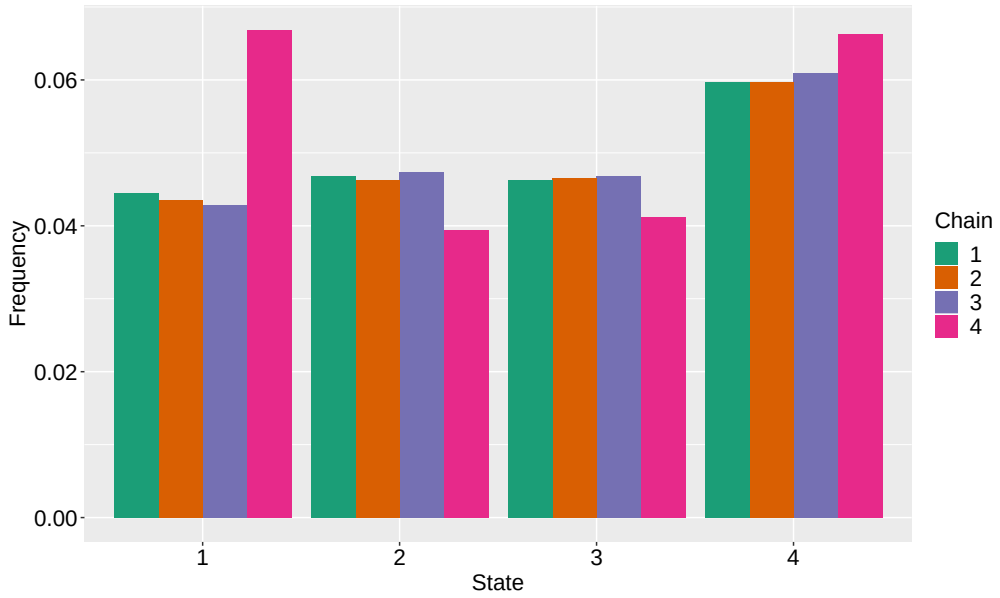

Figure 25: **Large state-space discrete distribution example: approximate stationary distribution.** The distribution over the first four states of a 20-state discrete model achieved after drawing 100,000 samples from each of four Markov chains. Here, the first three chains shared the same transition probability matrix; the last chain used a different matrix.

To probe the ability of R^* and $\widehat{R}$ to detect poor convergence, we repeat the same exercise as in §10.1 although, here, we consider the new $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ cases. In Fig. 26, we show the results of this analysis. This shows that $R^* \sim 1$ for the $\boldsymbol{P}_1$ case, correctly indicating no convergence issues; for the $\boldsymbol{P}_2$ case, $R^* > 1$ across all replicates at sample sizes of 5000 and above. By contrast, $\widehat{R}$ struggles to differentiate between the $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ cases: in both instances, it signifies convergence.

Collectively, the results of §10.1 and §10.2 show that R^* is able to detect convergence issues in discrete target distributions. The results of §10.2 hint that R^* may be superior than $\widehat{R}$ in larger discrete spaces, although further empirical investigation is needed.

11 ML sensitivity

In this section, we investigate how two decisions about classifiers which classifier to use (in §11.1) and what hyperparameters to use for it (in §11.2) affect calculation of R^*. The test cases used for empirical evaluation were selected from our pool of examples

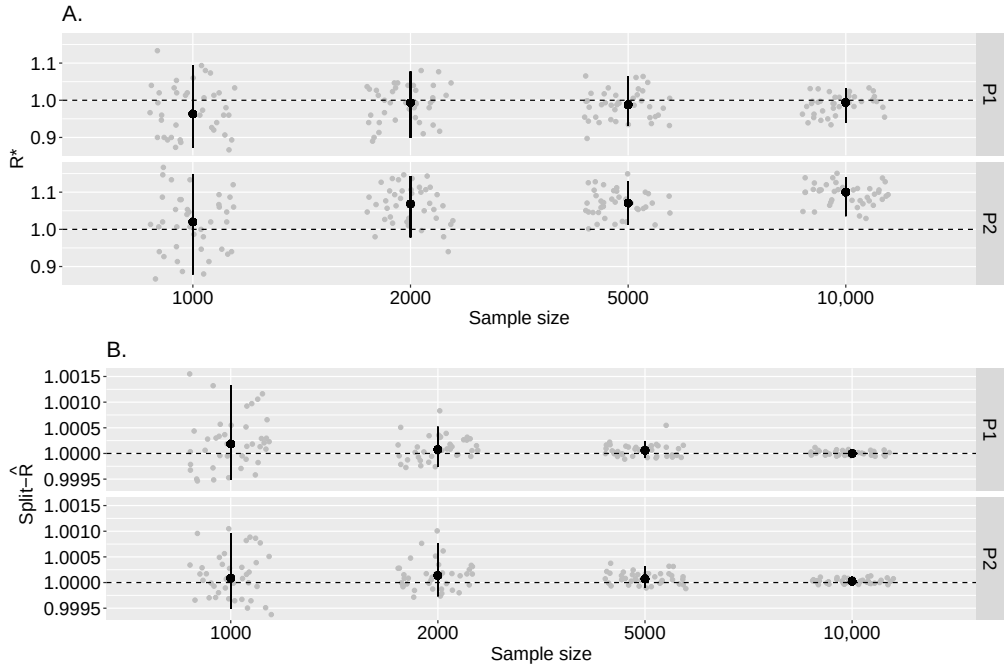

Figure 26: **Large state-space discrete distribution example: convergence diagnostics.** On the horizontal axis, we show the sample size of each set of simulations. The panels within A. and B. indicate the transition probability matrix used for the fourth chain (the other three chains always used $\boldsymbol{P} = \boldsymbol{P}_1$). In panel A, grey points indicate R^* from each replicate calculated using Algorithm 1) , and the black point-ranges indicate 2.5% , 50% and 97.5% quantiles across the 40 replicates. In panel B, we show the same, but for split-$\widehat{R}$. The dashed horizontal lines at 1 indicate the convergence threshold in both cases.

introduced in the text. Specifically, they were chosen to represent cases where R^* was nearer 1, since, for these, small changes in calculated R^* could lead to different decisions about convergence being reached. The four examples we used were:

- The AR(1) example described in §3.1, except using 1000 draws per chain.
- The 250-dimensional non-centered multivariate normal example introduced in §3.2. In each replicate, we generated 500 post-warm-up draws from the posterior (using 500 warm-up steps) using Stan's NUTS algorithm across each of 4 chains.
- The alternative parameterisation of the Cauchy model described in §3.3. For each replicate, we generated samples from the model using Stan's NUTS algorithm with 1000 post-warm-up draws (and 1000 warm-up draws) across each of 4 chains.

- The non-centered parameterisation of the eight schools model described in §3.4. In each replicate, we used Stan's NUTS algorithm with adapt delta set to 0.95 to generate 1000 post-warm-up draws (and 1000 warm-up draws) across each of 4 chains; these were then used to calculate R^*.

11.1 ML classifier comparison

In this section, we investigate how classification accuracy depends on choice of ML classifier. We restricted our analysis to popular classifiers with relatively few hyperparameters to simplify comparison amongst them. Notably, we do not consider neural network models, since the structure of nets effectively defines a high dimensional hyperparameter space.

The ML classifiers we compared were GBMs, RFs, k-nearest neighbours (KNNs), SVMs with linear kernels and a generalised linear model approach (GLM). All these models were called through the **R**'s Caret package (Kuhn et al., 2008): the GBM is implemented using the *gbm* model in Caret which uses the "gbm" package (Greenwell et al., 2019); RFs are implemented using the *rf* model in Caret which uses the "randomForest" (Liaw and Wiener, 2002) package; KNN is natively implemented in Caret and called using the *knn* model; SVMs are implemented in Caret using the *svmLinear* model which uses "kernlab" package (Karatzoglou et al., 2004); and the GLM is implemented through the *multinom* model in the "nnet" package (Ripley et al., 2016).

Each of these classifiers has hyperparameters, and, due to the differences between the examples we consider (defined in §11), the optimal hyperparameters were likely to differ between examples. For each classifier-example combination, we first performed a single replicate of each experiment and choose those hyperparameters which maximised classification accuracy. The set of all hyperparameters we selected between were chosen so that training time for each classifier remained reasonable. These sets were the same across all examples:

- GBM: Cartesian product of int.depth $= (3, 7)$; # trees $= (50, 100)$;
- RF: $m_{\text{try}} = (1, 2)$. (Note, in §11.2, we investigate a dimension-specific approach to m_{try} but, here, for ease of comparison with the other methods, we keep this static.)
- KNN: $K = (5, 10, 15, 20, 40)$;
- SVM: $C = (0.25, 0.5, 0.75)$;
- GLM: decay $= (0.1, 0.2, 0.5, 1)$.

These hyperparameters were then used in subsequent replicates. We also used a different number of replicates at each unique combination of classifier and hyperparameters across the experiments due to the differing demands of each example: for the

AR(1) example, we used 100 replicates; for the multivariate normal example, we used 100; for the Cauchy model, 50 replicates; and for the eight schools model, 30 replicates.

The results of these experiments are shown in Fig. 27. The top row shows R^* as calculated by each classifier (in each case, chains split into two halves); the bottom row shows the time (in seconds) taken for R^* calculation. The panels correspond to the examples described in §11. Across the examples, the classification accuracy of RFs and GBMs were, generally, highest; followed by KNNs. An exception was for the 250-dimensional normal where KNN performed best. In the AR(1) example, GBM outperformed RF. In higher dimensional cases, RF outperformed GBM. In all cases, RF and GBM classifiers took the greatest time to train; in higher dimensional examples, RF-based R^* were cheaper to calculate than those from GBMs.

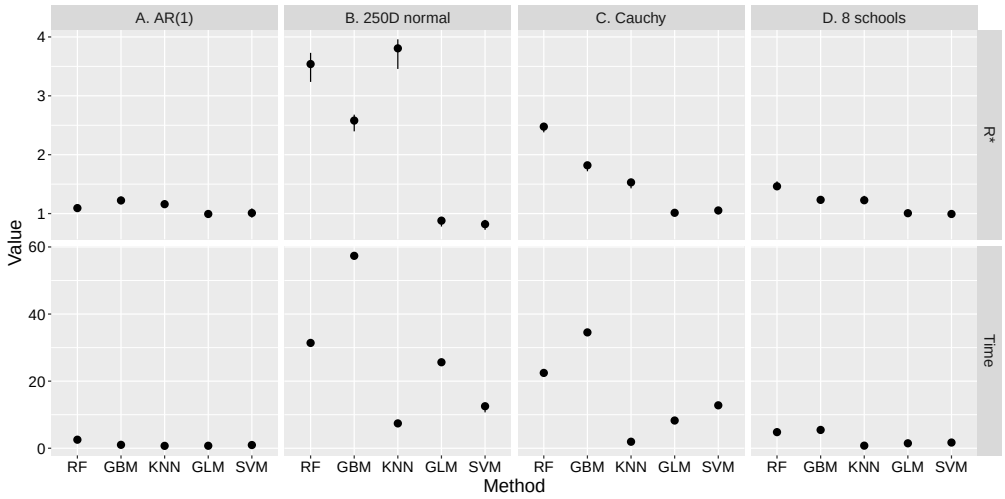

Figure 27: **ML classifier comparison.** Top row shows R^* values calculated using Algorithm 1; bottom row show the time taken to run each method on a desktop computer. The different columns correspond to the examples described in §11. Black point-ranges indicate 25% , 50% and 75% quantiles across all replicates; note, that the number of replicates varies across examples as given in §11.1.

11.2 Hyperparameter sensitivity

In this section, we considered only the two classifiers GBMs and RFs that performed best on our comparison between ML methods in §11.1. The performance of GBMs and RFs, like most ML methods, depends on their hyperparameters. In GBMs, a regression function is approximated by additive components, where each of those corresponds to a tree. The number of such regression trees is thus a hyperparameter of the model. Each regression tree can split amongst any integer number of variables (the *interaction depth*). For GBMs, we analyse how their performance depends on these two hyperparameters.

For RFs, there is a single hyperpameter: m_{try}, the number of members of the subset of all features over which to search for an optimal split when forming a decision tree. Note, that the maximum allowed value of $m_{\text{try}} = K$ the dimensionality of the target distribution.

In this section, we investigate the sensitivity of R^* calculated by GBMs and RFs to each of these sets of hyperparameters across a range of examples (defined in §11). Since each example is of different dimensionalities, we compare different sets of hyperparameters in each, although include amongst the sets the default values we suggest for each classifier in §2 (these are shown as triangles on the plots). Note, that for the GBM model, we did not investigate a full range of hyperparameters needed to optimise R^* for many of the examples due to the extensive training time needed to do so. For RFs, runtime was less restrictive and we were better able to survey the sensitivity across hyperparameter space.

We also used a different number of replicates at each unique combination of classifier and hyperparameters across the experiments due to the differing demands of each example: for the AR(1) example, we used 200 replicates; for the multivariate normal example, we used 20; for the Cauchy model, 20 replicates; and for the eight schools model, 50 replicates.

Autoregressive example

In Fig. 28, we show the results of the AR(1) analysis. In panel A, we show the sensitivity of R^* as calculated by GBMs to variation in hyperparameters. In panel B, we show the same but for RFs. In both panels, the results for the parameterisation we suggest as defaults is shown as a triangle. For the GBM model, R^* varied according to hyperparameters: of the set investigated, an interaction depth of 1 appears optimal with 10+ trees. In panel B, the values of R^* are generally lower than those achieved by GBMs. As shown in §12, RFs appear less well-suited to lower dimensional problems. Across the two possible hyperparameter values for RFs, there was not a substantial difference in the values of R^* calculated.

Multivariate normal: 250-dimensional model

In Fig. 29, we show the results of the analysis for the 250-dimensional normal example. In panel A, we show the sensitivity of R^* as calculated by GBMs to variation in hyperparameters. In panel B, we show the same but for RFs. This shows that R^* was higher as calculated by RFs and that performance of GBMs depended strongly on hyperparameters.

Cauchy model: alternative parameterisation

In Fig. 30, we show the results of the Cauchy model analysis. In panel A, we show R^* as calculated by a GBM classifier; in panel B, the same but using a RF classifier. In both panels, the results for the parameterisation we suggest as defaults is shown as a triangle.

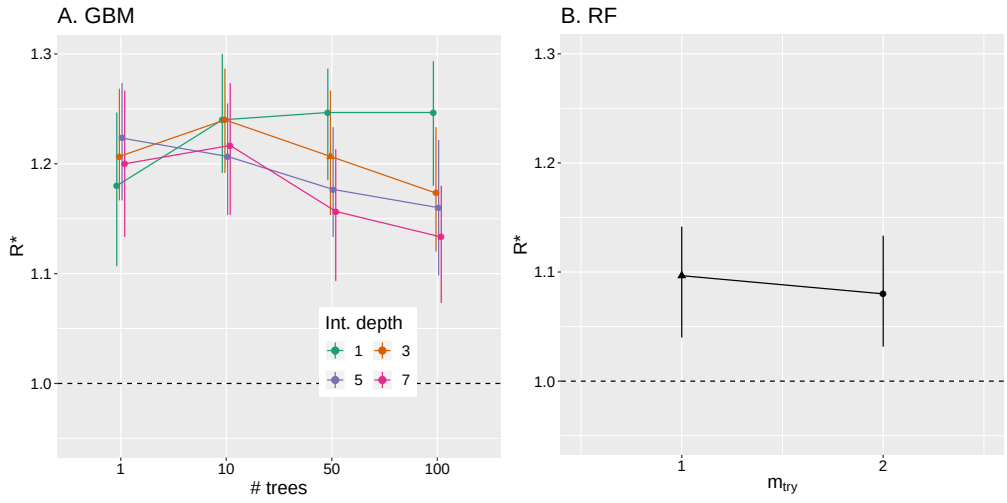

Figure 28: **Hyperparameter sensitivity: AR(1) example.** Panel A shows the sensitivity of GBM-derived R^* to hyperparameter variation for the AR(1) example described in §11.2; Panel B shows the same but for a RF classifier. Black point-ranges indicate 25% , 50% and 75% quantiles across all replicates. In Panel A, the coloured lines shows the different interaction depths investigated. In both panels, the results for the parameterisation we suggest as defaults is shown as a triangle.

Across the range of hyperparameters investigated, the RF classifier outperformed the GBM and was less sensitive to variation in its hyperparameters.

Eight schools: non-centered parameterisation

In Fig. 31, we show the results of the experiments on the eight schools model. Panel A shows how R^* calculated using a GBM varies with its two hyperparameters; panel B shows the same but for RFs. This shows that, in this case, the value of R^* calculated by RFs was higher than that for GBMs. Further, whereas GBMs were relatively sensitive to their hyperparameter values, RFs were less so.

12 Comparing GBMs and RFs

In §11, we present a series of experiments using popular ML methods and find that, among these methods, GBMs and RFs perform most consistently across them. In this section, we compare R^* derived from using a GBM classifier with that from a RF classifier. In these experiments, we fix the hyperparameters of each as given in §2. In §12.1, we compare how both methods are able to diagnose lack of convergence in a joint distribution. In §12.2, we compare the ability of both approaches to diagnose differences in the tails of the marginal distributions between chains. Since we use independent

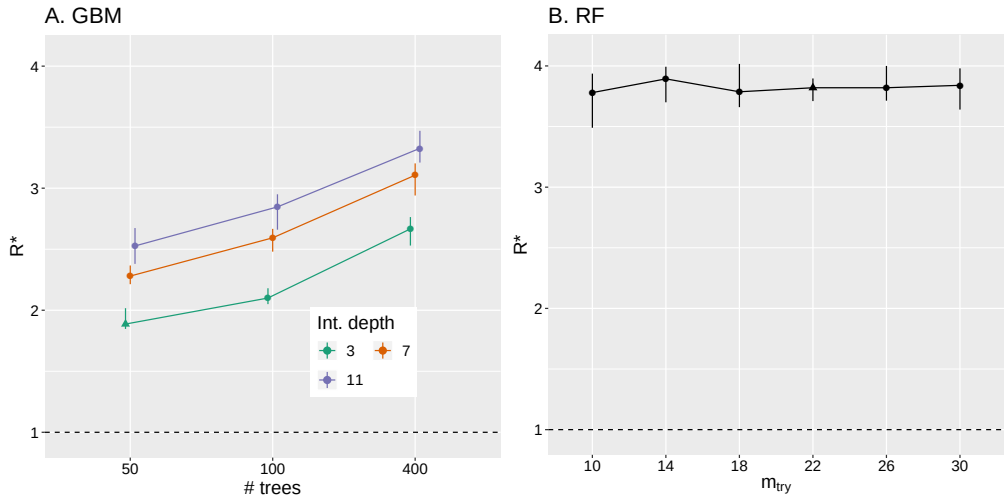

Figure 29: **Hyperparameter sensitivity: 250-dimensional normal example.** Panel A shows the sensitivity of GBM-derived R^* to hyperparameter variation for the Cauchy example described in §11.2; Panel B shows the same but for a RF classifier. Black point-ranges indicate 25% , 50% and 75% quantiles across all replicates. In Panel A, the coloured lines shows the different interaction depths investigated. In both panels, the results for the parameterisation we suggest as defaults is shown as a triangle.

sampling to generate draws, we are also able to calculate the *Bayes optimal* classification accuracy, which is the classifier that minimises the probability of misclassification error (Devroye et al., 2013). This classifier assigns chain identities according to the one which has the maximum posterior probability of having generated a draw. Here, we assume that *a priori* all chains are equally likely to have generated the data, so the classifier categorises draws to the chain which maximises the likelihood of that observation.

12.1 Joint distribution

Here, we consider a multivariate normal target distribution with dimensionality ranging from 1 to 32. For three 'chains', we generated 2000 draws by independently sampling from a multivariate normal with zero mean and given covariance matrix; for the fourth chain, the same number of draws are generated in the same way except with a different covariance matrix. To generate the covariance matrices, we randomly sampled two 32-dimensional covariance matrices from the LKJ distribution with degrees of freedom 1 (Lewandowski et al., 2009): note, that this results in matrices with unit diagonal values, meaning the marginal distributions of each dimension are standard normals. To obtain covariance matrices for the targets with dimensions, $d < 32$, submatrices, $A^{\{d\}}$, were constructed from the 32-dimensional covariance matrices by taking leading blocks from them: $A^{\{d\}} := A^{\{32\}}_{1:d,1:d}$. For each target, we performed 20 replicates, where in each case, R^* was calculated using GBM and RF classifiers. In addition, we determined an

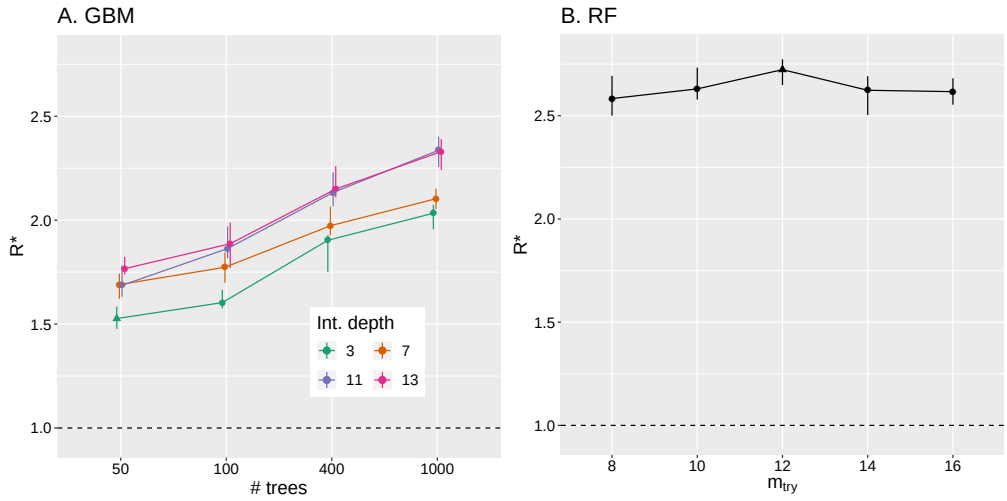

Figure 30: **Hyperparameter sensitivity: Cauchy example.** Panel A shows the sensitivity of GBM-derived R^* to hyperparameter variation for the Cauchy example described in §11.2; Panel B shows the same but for a RF classifier. Black point-ranges indicate 25% , 50% and 75% quantiles across all replicates. In Panel A, the coloured lines shows the different interaction depths investigated. In both panels, the results for the parameterisation we suggest as defaults is shown as a triangle.

R^* using the Bayes optimal classifier: to do so, we generated 10,000 draws from the four-chain process and assign chain identities to the maximum likelihood class.

The results of these experiments are shown in Fig. 32. In this plot, the bottom axis shows the dimensionality of the target distribution; the vertical axis shows R^*, and the point colour and shadings give the method used to calculate this statistic. When the target is unidimensional, the four chains have the same target distribution and the optimal $R^* = 1$. In this case, both the GBM and RF methods produce classification rates that overlap with this value, indicating convergence. As the number of dimensions increases, it becomes easier to differentiate between samples from the two processes and the optimal R^* grows. R^* as calculated by GBMs and RFs also increases. For a two-dimensional target, the GBM method outperforms the RF one, getting closer to optimal classification. For higher dimensional targets, the RF classifier outperforms, achieving near-optimal R^* in 32 dimensions.

12.2 Fat tails

We now compare how the GBM and RF methods are able to diagnose lack of convergence in the tails of a target distribution. To do so, we again consider four chains. In three of these chains, we generated 2000 draws by randomly sampling from a multivariate Student-t distribution with mean zero and 3 degrees of freedom. In the remaining chain,

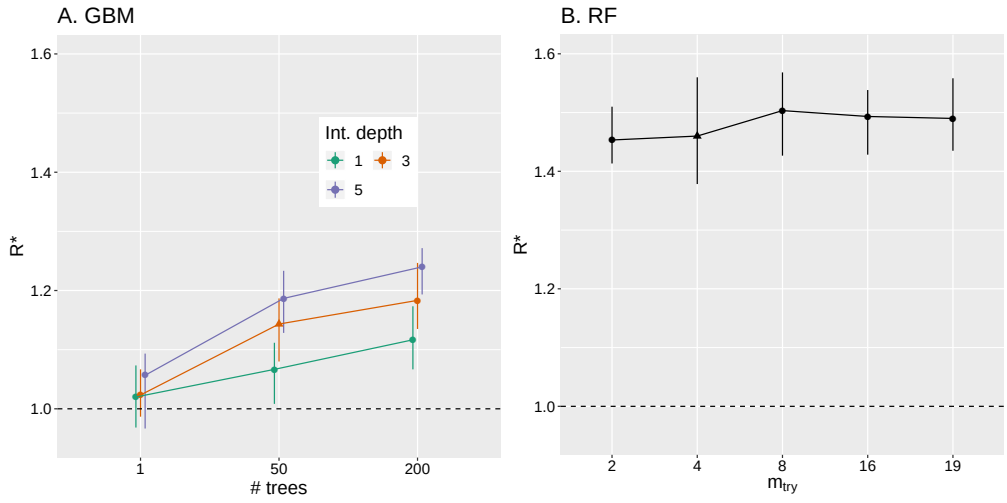

Figure 31: **Hyperparameter sensitivity: eight schools example.** Panel A shows the sensitivity of GBM-derived R^* to hyperparameter variation for the eight schools example described in §11.2; Panel B shows the same but for a RF classifier. Black point-ranges indicate 25% , 50% and 75% quantiles across all replicates. In Panel A, the coloured lines shows the different interaction depths investigated. In both panels, the results for the parameterisation we suggest as defaults is shown as a triangle.

we generated the same number of independent draws but use a mean-zero multivariate Student-t distribution with differing degrees of freedom, ν. We consider a range of target dimensions: to generate the shape matrix for the first three chains, we use the same approach as in §12.1 to build up appropriate shape matrices for each target dimensions, $A^{\{d\}}$. We examine a range of values of $\nu = \{4, 8, 16, 32\}$, representing decreasing tail fatness. Across the range of ν considered, the covariance matrix for the first three chains is given by $\nu/(\nu-2)A^{\{d\}}$. To ensure that the fourth chain has the same covariance as the first three, we use a shape matrix $(\nu-2)/\nu A^{\{d\}}$. For each combination of dimensions and ν, we performed 20 replicates, where in each, we again calculate R^* for both the GBM and RF classifiers. Additionally, we estimate an optimal R^* using the Bayes optimal classifier by calculating the predictive accuracy of 10,000 draws of the four-chain process.

The results of these experiments are shown in Fig. 33. Here, each panel shows a target distribution of different dimensionality: 1, 2, 4, 8, 16 and 32 dimensions. Within each panel, the horizontal axis shows the degrees of freedom, ν, of the fourth chain; the vertical axis gives R^*. The point colour and shadings give the method used to calculate R^*. In each panel, increases in ν make it easier to differentiate draws from the fourth chain from those of the first three: accordingly R^* tends to increase for each method. As the target dimensionality increases (panels left-right along each row), it also becomes easier to contrast draws from the fourth chain, and R^* increases. In all cases, the optimal R^* exceeds those using GBM or RF classifiers. In 1-4 dimensions, the GBM approach

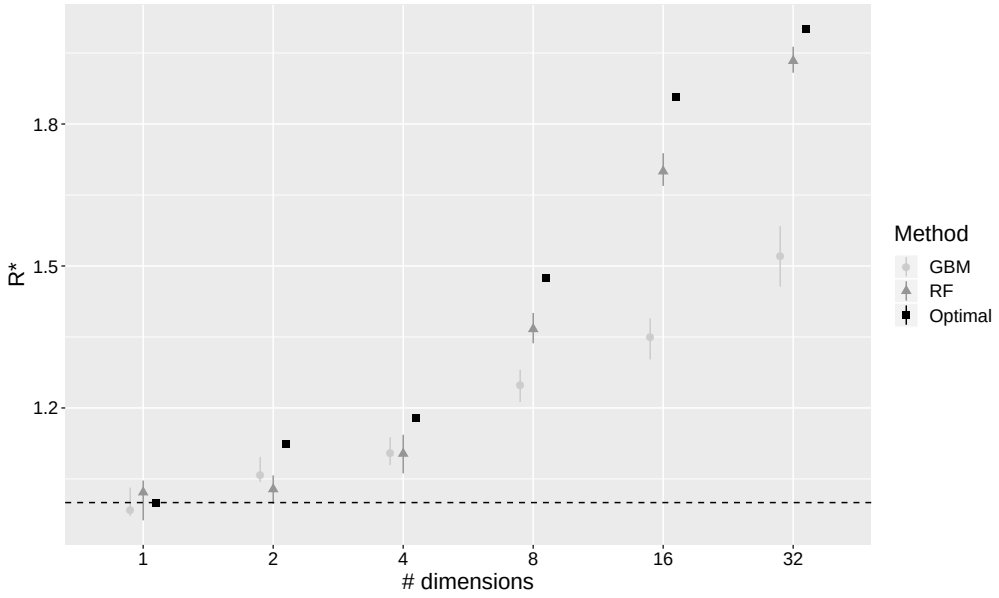

Figure 32: **GBM versus RF: multivariate normal.** The horizontal axis gives the dimensions of the target distribution; the vertical axis gives the value of R^* as calculated by Algorithm 1 using a GBM (light grey points), a RF (dark grey triangles) and the Bayes optimal classifier (black squares). The upper and lower point ranges show the 75% and 25% quantiles and the filled shapes show the medians. For the Bayes optimal classifier, there is only a point estimate since the optimal accuracy was estimated by Monte Carlo sampling using 10,000 draws from the four chain process. The dashed line shows $R^* = 1$. Here, R^* was calculated using chains split into two halves.

outperforms the RF one, getting closer to the optimal classification rate. If the number of dimensions is 16 or greater, the order switches, and RFs often outperform GBMs.

Supplementary materials: R^* convergence diagnostic. 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. 3, 6, 23

Betancourt, M. (2017). "A conceptual introduction to Hamiltonian Monte Carlo." *arXiv preprint arXiv:1701.02434*. 1, 13, 19

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–

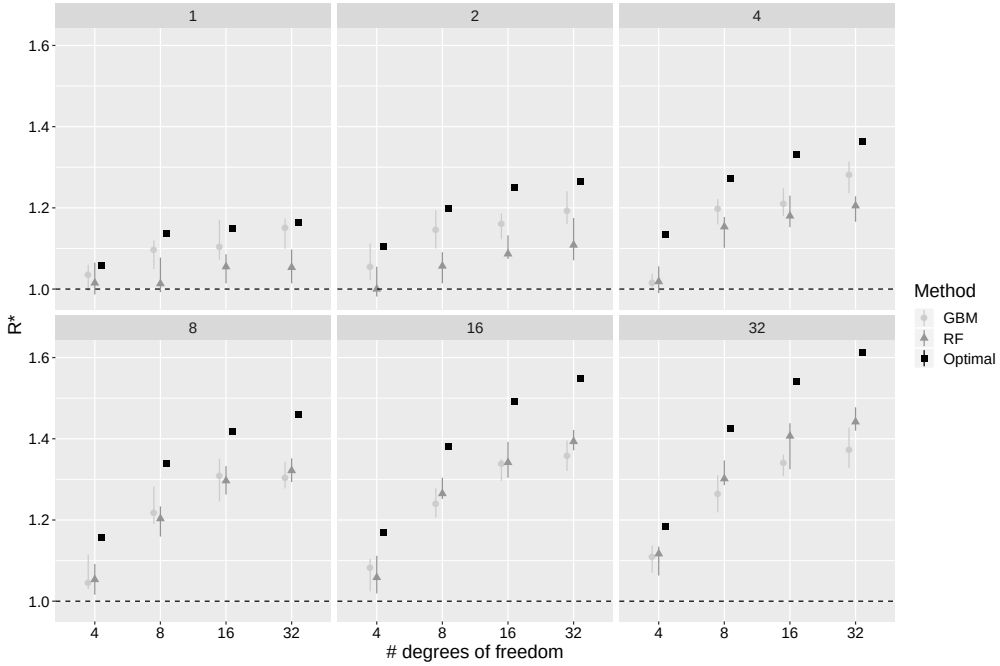

Figure 33: **GBM versus RF: multivariate Student-t.** Different panels show the various dimensions of the target distribution. The horizontal axis gives the degrees of freedom of the multivariate Student-t distribution for the fourth chain; the vertical axis gives the value of R^* as calculated by Algorithm 1 using a GBM (light grey points), a RF (dark grey triangles) and the Bayes optimal classifier (black squares). The upper and lower point ranges show the 75% and 25% quantiles and the filled shapes show the medians. For the Bayes optimal classifier, there is only a point estimate since the optimal accuracy was estimated by Monte Carlo sampling using 10,000 draws from the four chain process. The dashed line shows $R^* = 1$. Here, R^* was calculated using chains split into two halves.

978. 1

Boehmke, B. and Greenwell, B. (2019). *Hands-on machine learning with R*. CRC Press. 3, 5, 6

Breiman, L. (2001). "Random forests." *Machine Learning*, 45(1): 5–32. 3, 4

Brooks, S., Gelman, A., Jones, G., and Meng, X. (2011). *Handbook of Markov chain Monte Carlo*. CRC press. 2

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. 18

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). 1, 2, 19, 29

Devroye, L., Györfi, L., and Lugosi, G. (2013). *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media. 23, 46

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*. 1

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` 17

Friedman, J. (2001). "Greedy function approximation: a gradient boosting machine." *Annals of Statistics*, 1189–1232. 3, 4, 14

Ge, H., Xu, K., and Ghahramani, Z. (2018). "Turing: A language for flexible probabilistic inference." 1

Gelman, A. and Rubin, D. (1992a). "Inference from iterative simulation using multiple sequences." *Statistical Science*, 7(4): 457–472. 2

— (1992b). "A single series from the Gibbs sampler provides a false sense of security." *Bayesian Statistics*, 4: 625–631. 2

Gelman, A., Stern, H., Carlin, J., Dunson, D., Vehtari, A., and Rubin, D. (2013). *Bayesian data analysis*. Chapman and Hall/CRC. 2, 18, 27

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. 1

Greenwell, B., Boehmke, B., Cunningham, J., Developers, G., and Greenwell, M. B. (2019). "Package gbm." 3, 5, 14, 42

Hernández-Lobato, D., Hernández-Lobato, J., and Suárez, A. (2010). "Expectation propagation for microarray data classification." *Pattern Recognition Letters*, 31(12): 1618–1626. 33

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. 1, 13

Karatzoglou, A., Smola, A., Hornik, K., and Zeileis, A. (2004). "kernlab-an S4 package for kernel methods in R." *Journal of Statistical Software*, 11(9): 1–20. 42

Kuhn, M. et al. (2008). "Building predictive models in R using the Caret package." *Journal of Statistical Software*, 28(5): 1–26. 5, 42

Lambert, B. (2018a). *A Student's Guide to Bayesian Statistics*. Sage Publications Ltd. 2

— (2018b). "YouTube video: Bobs bees: the importance of using multiple bees (chains) to judge MCMC convergence." 2

Lewandowski, D., Kurowicka, D., and Joe, H. (2009). "Generating random correlation matrices based on vines and extended onion method." *Journal of Multivariate Analysis*, 100(9): 1989–2001. 46

Liaw, A. and Wiener, M. (2002). "Classification and regression by randomForest." *R news*, 2(3): 18–22. 5, 42

Louppe, G. (2014). "Understanding random forests: From theory to practice." *arXiv preprint arXiv:1407.7502*. 25

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. 1

Neal, R. et al. (2011). "MCMC using Hamiltonian dynamics." *Handbook of Markov chain Monte Carlo*, 2(11): 2. 1

Paananen, T., Piironen, J., Brkner, P., and Vehtari, A. (2019). "Implicitly Adaptive Importance Sampling." *arXiv*. 33

Piironen, J. and Vehtari, A. (2017). "Sparsity information and regularization in the horseshoe and other shrinkage priors." *Electronic Journal of Statistics*, 11(2): 5018–5051. 33

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. 1

Ripley, B., Venables, W., and Ripley, B. (2016). "Package nnet." *R package version*, 7: 3–12. 42

Salvatier, J., Wiecki, T., and Fonnesbeck, C. (2016). "Probabilistic programming in Python using PyMC3." *PeerJ Computer Science*, 2: e55. 1

Schummer, M., Ng, W., Bumgarner, R., Nelson, P., Schummer, B., Bednarski, D., Hassell, L., Baldwin, R., Karlan, B., and Hood, L. (1999). "Comparative hybridization of an array of 21,500 ovarian cDNAs for the discovery of genes overexpressed in ovarian carcinomas." *Gene*, 238(2): 375–385. 33

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., Tamayo, P., Renshaw, A., D'Amico, A., and Richie, J. (2002). "Gene expression correlates of clinical prostate cancer behavior." *Cancer Cell*, 1(2): 203–209. 33

Van Dyk, D. and Meng, X. (2001). "The art of data augmentation." *Journal of Computational and Graphical Statistics*, 10(1): 1–50. 18

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*. 2, 4, 7, 12, 14, 15, 16, 18, 19, 22, 28, 29

Wolpert, D. and Macready, W. (1997). "No free lunch theorems for optimization." *IEEE transactions on evolutionary computation*, 1(1): 67–82. 24

Yang, K., Cai, Z., Li, J., and Lin, G. (2006). "A stable gene selection in microarray data analysis." *BMC Bioinformatics*, 7(1): 228. 33

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.