How well does your sampler really work?

Ryan Turner
Uber AI Labs

Brady Neal
MILA, Université de Montréal

Abstract

We present a data-driven benchmark system to evaluate the performance of new MCMC samplers. Taking inspiration from the COCO benchmark in optimization, we view this benchmark as having critical importance to machine learning and statistics given the rate at which new samplers are proposed. The common hand-crafted examples to test new samplers are unsatisfactory; we take a meta-learning-like approach to generate realistic benchmark examples from a large corpus of data sets and models. Surrogates of posteriors found in real problems are created using highly flexible density models including modern neural network models. We provide new insights into the real effective sample size of various samplers per unit time and the estimation efficiency of the samplers per sample. Additionally, we provide a meta-analysis to assess the predictive utility of various MCMC diagnostics and perform a nonparametric regression to combine them.

1 INTRODUCTION

Markov chain Monte Carlo (MCMC) methods have seen a huge increase in use over the last few decades. The goal in MCMC methods is to take samples from a complex probability distribution $p^\star$ given access only to its unnormalized density $\tilde{p}$. The primary use case for MCMC methods is sampling from Bayesian posteriors for the purpose of Monte Carlo integration, which includes building posterior predictive distributions and posterior summaries. These posteriors are generally intractable to normalize and sample from in modern models, including models as simple as logistic regression. Approaches such as rejection sampling provide exact independent samples, and importance sampling provides exact independent (but weighted) samples. These approaches are generally computationally inefficient (rejection sampling) or are statistically unsound (importance sampling) except in very low dimensional problems [MacKay, 2003, Ch. 29]. MCMC methods produce a Markov chain that marginally samples from the target distribution $p^\star$ exactly and have a low per sample computation cost. The downside is that they provide a sequence of *correlated* samples, albeit marginally from the target distribution. Therefore, any estimates derived from an MCMC chain of length N will have far less accuracy than N iid samples. Despite there being numerous MCMC diagnostics, there is no practical way to guarantee the accuracy of derived estimates in practice.

Each machine learning conference contains a publication proposing a new variation on MCMC methods. The community lacks a method to determine if these new methods actually sample from posteriors found in real problems with improved accuracy over existing samplers. New methods are benchmarked via either 1) hand-crafted toy problems (where a ground-truth is known) or 2) test set performance on real problems. The issue with hand-crafted examples is obvious: Performance on these problems may have little relation to performance on real problems and it is at odds with accepted practice in modern machine learning.

Benchmarking via test set performance on real problems is laudable. However, it confounds the specification of the model and priors with the performance of the sampler. In a misspecified model it is possible that a sampler stuck in an unrepresentative part of the posterior could actually have higher test set performance [Sharp and Rattray, 2010]. Conversely, a better sampler may improve test set performance by having good local mixing; however, it is still nowhere near exact iid samples. There is no way to quantify the distance to exact iid samples from test set performance alone.

Whether current samplers are providing samples from anything close to the true posterior on difficult problems is of critical importance for determining future research directions. Are samplers with higher test set performance actually sampling from real posteriors more faithfully? Can we sample with any fidelity from complex high dimensional distributions? Is that merely a "fool's errand"? The answers to these questions will determine if it is a worthwhile endeavor to continue to hone MCMC methods for application in successful modern models such as deep neural networks.

Practitioners in Bayesian statistics have long faced the dilemma of whether they can trust the output of their sampler, in particular, because statisticians are not traditionally concerned only with test set error rates. As a result, there is decades of work in developing MCMC diagnostics that aim to *alert* a practitioner to a *poorly mixing* chain [Cowles and Carlin, 1996]. That is, if a chain has a long autocorrelation time, the entire chain may be of equivalent accuracy to just a few iid samples. The diagnostics, by construction, have a low type I error: If a chain closely resembles iid samples, they will not alert that it is mixing poorly. However, there are no guarantees on type II error: If a chain is mixing poorly, the diagnostic might not alert. Indeed, there are many ways to construct examples where an MCMC procedure undetectably fails: distant modes, Neal's funnel [Thompson, 2011], extreme ill-conditioning, etc. However, are these realistic stress tests for MCMC methods or merely pathological cases? We do not know.

We propose a new data-driven approach to create a benchmark that estimates how well various MCMC procedures work on real problems. Arguably, algorithms in machine learning and statistics rely on the "workhorses" of either optimization or sampling methods. The world of (non-convex) optimization has already tackled this challenge with the COCO benchmark [Hansen et al., 2016], which contains a test battery of difficult optimization problems. Various approaches are tested to validate if they can optimize the objective function to a target level within a fixed number of function evaluations. Our approach is an analogous system for sampling methods. However, we further improve upon this using flexible (including neural net based) benchmark examples that have been trained to match posteriors found in practice.

In our approach we use a large "data set of data sets" and a diverse "model zoo" to create a representative set of examples. Long MCMC chains are drawn (using NUTS [Hoffman and Gelman, 2014]) from each of these posteriors. Flexible unsupervised models that serve as a *ground-truth* in the benchmarking phase are fit to the chains to construct the benchmark examples.

More concretely, each combination of real data set (e.g., MNIST) and real model (e.g., logistic regression) results in a Markov chain from NUTS. We then fit an unsupervised model (e.g., mixture of Gaussians) to this chain to serve as a *benchmark example distribution*. Once trained, these benchmark example distributions are functionally equivalent to hand-crafted examples such as the toy posterior distributions usually used to benchmark samplers (or such as those in COCO). However, these examples are not hand-crafted but rather are much more representative of real problems. Because it is possible to draw exact (iid) samples from the benchmark example distributions, we now have a ground-truth set of samples to validate the accuracy of the sampling methods.

We derive a variety of metrics that summarize the performance of a sampler for comparing its output to ground-truth iid samples. The ground-truth samples also allow us to assess how well the MCMC diagnostics actually predict estimation performance. In particular, we look at the effective sample size (ESS) because it provides a concrete statement on sample quality [Kass et al., 1998].

The outline of this paper is as follows: In Section 2 we provide some background on MCMC and its diagnostics/performance measures. In Sections 3 and 4 we explain the methodology of the benchmark and its pipeline of five sequential phases. Finally, in Section 5, we present results illustrating the advantages of various samplers and the utility of various MCMC diagnostics.

Contributions We summarize the contributions of this work as follows: 1) We provide a new and novel benchmark to describe how well various samplers work on realistic problems. This involves design of fair and sensible metrics to score samplers across problems. This work creates a software system that will serve as a practical tool in algorithm development analogous to MLcomp/CodaLab or COCO. 2) We shed light on how well the common MCMC diagnostics predict the real estimation performance of MCMC methods. We further create a data-driven meta-diagnostic by combining MCMC diagnostics to predict real sampler performance. The code for the system is available at `github.com/bradyneal/sampling-benchmark`.

Related work The closest existing system is SamplerCompare of Thompson [2011], which tests samplers on a handful of hand-crafted stress-test cases such as Neal's funnel. However, SamplerCompare is more an R package to aid evaluation than a complete benchmark. A recent piece of work from systems biology [Ballnus et al., 2017] compares various samplers for dynamical systems (i.e., filtering) on a set of hand-crafted ODE systems inspired by biological models.

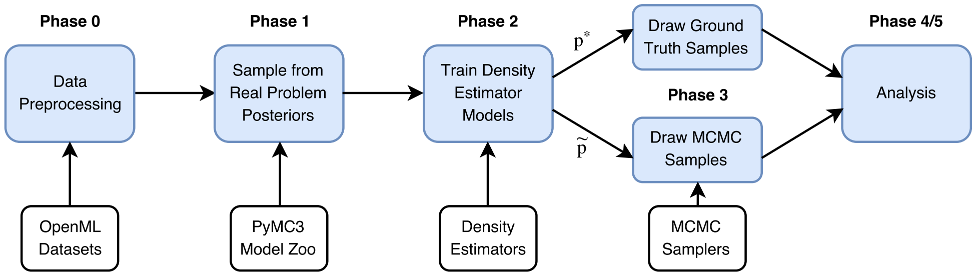

Figure 1: Flowchart illustrating the six phases in our methodology. Phases 0–2 are for creating benchmark examples and are not re-run when new samplers are tested. Phase 2 includes mixture models and modern neural net methods.

2 BACKGROUND

The notion of a black box is highly relevant to conceptual understanding of this work. Fundamentally, an MCMC sampler is a system that inputs a black box that computes an unnormalized density $\tilde{p} \propto p^\star$ (and possibly its gradient $\nabla \log \tilde{p}$) and a previous sample $\mathbf{x}_{t-1} \in \mathbb{R}^D$ in the Markov chain; and outputs another sample $\mathbf{x}_t \in \mathbb{R}^D$. Once the Markov chain has converged, these samples are theoretically guaranteed to marginally come from the density $p^\star$, albeit with temporal correlation. If the previous sample was drawn exactly, $\mathbf{x}_{t-1} \sim p^\star$, then $\mathbf{x}_t \sim p^\star$ exactly as well. This is a result of *detailed balance*.

By analogy, optimization algorithms take an objective function $f \in \mathbb{R}^D \to \mathbb{R}$ (and possibly its gradient ∇f) as a black box and produce points $\mathbf{x}_t \in \mathbb{R}^D$ that successively minimize f as much as possible. Just as COCO provides its benchmark objective functions f as a black box to the optimizers and keeps hidden the true optimum, our benchmark provides the unnormalized density $\tilde{p}$ as a black box to the samplers. Our benchmark keeps hidden the parameterization of $\tilde{p}$ needed to efficiently take iid samples from $p^\star$.

2.1 TRADITIONAL MCMC DIAGNOSTICS

Given that we have a ground-truth to evaluate the performance of the various samplers, we can also benchmark the diagnostics by seeing how predictive they are of actual performance. In particular, we consider three diagnostics in this paper: ESS, Gelman-Rubin (GR), and Geweke. ESS aims to estimate how many iid samples have the same estimation performance as the correlated samples found in the MCMC chain. Gelman-Rubin [Gelman and Rubin, 1992] and Geweke [Geweke, 1992] more closely follow a test statistic paradigm than an estimation one. Gelman-Rubin compares the variance within a single chain to variance between chains (independent restarts). This quantity should be close to one for well-mixing chains and can be very large for poorly performing chains. The Geweke diagnostic uses a single chain and compares the variance between chunks.

The ESS diagnostic is basically a rescaling of the expected square error (i.e., MSE) on estimating the mean in a *single dimension* (marginal) of $\mathbf{x}$. ESS is based on the notion that for the marginal x_d:

$$\begin{aligned} \mathbb{E}_{p^\star}[(\hat{\mu}_d - \mu_d)^2] &= \mathrm{Var}_{p^\star}[\hat{\mu}_d - \mu_d] + \mathbb{E}_{p^\star}[\hat{\mu}_d - \mu_d]^2 \\ &= \mathrm{Var}_{p^\star}[x_d]/N\,, \quad d \in 1{:}D\,, \qquad (1) \\ \hat{\boldsymbol{\mu}} &:= \tfrac{1}{N}\textstyle\sum_{i=1}^N \mathbf{x}_i\,, \quad \boldsymbol{\mu} := \mathbb{E}_{p^\star}[\mathbf{x}]\,, \qquad (2) \end{aligned}$$

which utilizes that $\hat{\mu}_d$ is an unbiased estimate of μ_d. We are careful to distinguish expectations and variances with respect to $p^\star$, where $\mathbf{x}$ is iid, from q, where the samples are correlated and drawn from an MCMC method. Naturally, by re-arranging (1), the effective sample size for non-iid samples is:

$$\mathrm{ESS} := \frac{\mathrm{Var}_q[x_d]}{\mathbb{E}_q[(\hat{\mu}_d - \mu_d)^2]} \in \mathbb{R}^+\,. \qquad (3)$$

Unlike (1), this can be estimated without ground-truth samples from $p^\star$. However, the difficult denominator term is typically estimated using the empirical linear auto-correlation of the Markov chain. This linearity assumption is obviously a potential source of error in the ESS. The fixation in estimating the accuracy of the mean $\hat{\mu}$ is also a weakness. In Section 4.6, we look at the *real* effective sample size by comparing estimates with the ground-truth samples. It also allows us to look at measures other than simply the fidelity in matching the means ($\hat{\mu} - \mu$), such as variance or shape of the marginals.

3 METHODOLOGY

Our benchmark system follows a six phase approach, which we explain at a high level in this section. In Section 4, we provide low-level specifics. A graphical summary of this section is provided in Figure 1.

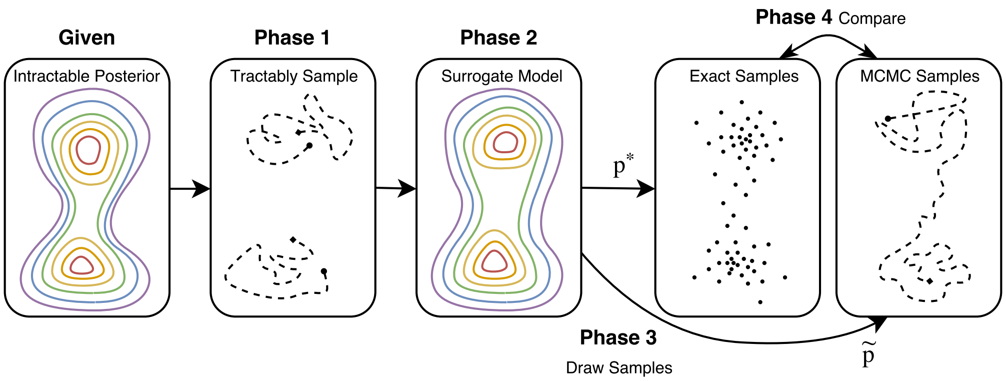

Figure 2: Graphical depiction of Figure 1 on a particular example. We begin with a real posterior from a real problem on the left, which is sampled via a Markov Chain to get samples in phase 1. These are fit to get a similar surrogate posterior in phase 2. MCMC samplers are run on this phase 2 density, but exact samples can also be taken for comparison. Note that the exact posterior, and the real data that produced it, *are not used* for comparing the exact samples and the MCMC samples in phase 4. The phase 2 models are used where toy examples are often used; although it is not the original posterior, it is a far-reaching improvement.

In phase 0, we create a "corpus" of data sets that we refer to as a "data set of data sets." This is meant to create a realistic sample of problems that a practitioner may encounter "in the wild." Such an approach was also taken in the AutoML competition [Guyon et al., 2015] and the automated statistician project [Lloyd et al., 2014]. Our approach can be thought of as a form of *meta-learning*.

In phase 1, we use a model zoo to simulate a variety of (Bayesian) models that a practitioner might attempt to apply to a real problem. There are models for regression and classification. Each model/data set pair results in a posterior over a parameter space, which varies in dimensionality depending on the problem. Except in very simple cases (e.g., linear regression), we are not able to obtain samples from these posteriors exactly. We use NUTS, the default sampler in probabilistic programming languages (PyMC3 [Salvatier et al., 2016] and Stan [Carpenter et al., 2016]), because it is generally considered to be a good off-the-shelf sampler—especially when paired with the intelligent initialization and automatic tuning found in these systems. Therefore, by running multiple long chains of NUTS (3–5 chains for 30 minutes each) on the posteriors, we obtain a sufficient approximation and representation for phase 2.

In phase 2, we run various density estimation models to generate benchmark example distributions on the Markov chains from phase 1. We run a separate training procedure on each model/data set pair. These benchmark example distributions serve as surrogates for the real posteriors found in phase 1. Note that the goal is not to replicate the posteriors from phase 1 exactly, but to generate example distributions that are *qualitatively similar* to the real posteriors in phase 1. This gives us example distributions that are more realistic than the usual hand-crafted toy problems. Nonetheless, we train multiple models and take the one with the highest held-out likelihood on the last 20% of the Markov chain found in phase 1. We use held-out likelihood because it is the most widely accepted generic method of verifying model fidelity. Model checking diagnostics are also run to verify the similarity between the benchmark example distributions (surrogates) and their corresponding Markov chains from the real posteriors (originals).

When selecting models for benchmark example distributions in phase 2, we have the following requirements: 1) The models are flexible enough to closely fit the posteriors found in phase 1, 2) They can serve as a black box, providing an unnormalized density $\tilde{p}$ (and its gradient) when queried at an arbitrary point $\mathbf{x}$, and 3) We can efficiently sample (ground-truth) from them given their parameters (which are hidden from the samplers).

In phase 3, we benchmark a collection of samplers. If someone invents and provides a new sampling algorithm, it is added in phase 3. Phases 0–2 remain fixed as new samplers are submitted to be benchmarked. Each sampler to be benchmarked is run on each of the benchmark example distributions for multiple chains. Each chain is allowed to run for a fixed period of time. The samples from the Markov chains are saved as the phase 3 output.

In phase 4, we take a large number (e.g., $\sim 10^5$) of exact iid samples from the benchmark example distributions as a ground-truth. The square loss between point estimates (e.g., $\hat{\mu}_d$ or $\hat{\sigma}_d^2$) taken from the Markov chains from phase 3 and the point estimates from the exact chains are aggregated. We also compute and store the MCMC diagnostics for each chain.

In phase 5, we aggregate the performance results by looking at the real effective sample size as derived from the square errors in point estimation. We also define transformations of the real effective sample size, which we will refer to as efficiency, normalized effective sample size, and effective sample size deviation. In addition, we perform a meta-analysis using Gaussian process (GP) [Rasmussen and Williams, 2006] regression to predict the real effective sample size given the MCMC diagnostics. This will be useful to practitioners aiming to quantify their confidence in an MCMC-based estimate using the diagnostics available.

We present an example posterior following this pipeline in Figure 2. Note that after the explicit model is fit in phase 2, the data that produced the original posterior is completely *irrelevant* for the rest of the process. Only the surrogate model is used for benchmarking.

4 ADDITIONAL DETAILS

In this section we present additional details for the construction of each phase.

4.1 PHASE 0: COLLECT DATA SETS

Phase 0 involved downloading 2,200 data sets from `openml.org` to form our data set of data sets. We considered other sources, such as the classic UCI repository, `mldata.org`, and Kaggle, but settled on OpenML because it had the most standardized format and meta-data. Such systems are necessary for automated processing.

The data sets were diverse in that they varied in dimension from 1 to 61,359, sample size from 5 to 7,619,400, and the number of output classes (for classification) from binary to 100.

After downloading, we subjected each data set to some preprocessing to simulate the diverse set of practices a practitioner might follow. Each data set was randomly preprocessed in one of three ways: standardization, robust standardization (using medians and interquartile ranges), or whitening. Categorical variables were represented with one-hot encodings.

4.2 PHASE 1: SAMPLE THE MODEL ZOO

For the model zoo, we used all of the standard models (regression and classification) typically used with PyMC3. This includes generalized linear models (GLMs) such as logistic regression, but also atypical GLMs such as robust linear regression (linear regression with Student's-t noise). In addition to models that are linear in the feature space, we included models that are linear in a second order transformation of the feature space. We included Gaussian processes with unknown hyper-parameters (e.g., MCMC sampling was done on the unknown hyper-parameters). Bayesian neural networks were also included.

To keep computation time reasonable, we limited the sample size for expensive models (e.g., GPs), and placed some limits on input dimensionality. Where dimensionality needed to be reduced we used PCA [Jolliffe, 1986], as that is the most frequently used method in practice to reduce dimensionality.

4.3 PHASE 2: FIT FLEXIBLE SURROGATES

There are three varieties of models that satisfy the three requirements (flexibility, tractable density, and fast exact sampling) for benchmark example densities: mixture models, RNADE [Uria et al., 2013], and Real NVP [Dinh et al., 2016]. In each example, we pick the model with the best held-out likelihood on the last 20% of the chain.

For mixture models, we considered mixture of Gaussians (MoG) with expectation-maximization (EM) [Dempster et al., 1977] and variational MoG. Note that, for simplicity, these models are *not* themselves fit using MCMC. The Bayesian Occam's razor effect [Jefferys and Berger, 1992] allowed us to simply fix the number of mixture components to 25 in variational MoG. We used five-fold cross-validation to select the number of components in EM MoG. There is no consistent winner between these models; the chosen model is example dependent.

We also tuned the RNADE learning rate and hyper-parameters based on pilot runs. Surprisingly, the mixture models often, but not always, outperformed RNADE on the held-out likelihood. Real NVP based models struggled to achieve competitive test set scores.

These models behave better numerically when trained on standardized data. Care is taken to reverse this standardization in phase 3, so the samplers are forced to attempt to sample from the posterior in its original scale.

4.4 PHASE 3: RUN THE SAMPLERS

Phase 3 forms the real "meat" of the benchmark. This is where candidate sampling algorithms are actually run on the benchmark example densities. The list of sampling algorithms is not intended to be exhaustive but rather demonstrate the utility of the benchmark system.

Whether originally designed this way or not, nearly all respected MCMC procedures proceed by proposing a new point using a *proposal distribution*. The new point is then accepted or rejected using a Metropolis-Hastings step. Therefore, the difference between samplers is based upon their proposal distributions. We provide a preview of the proposals used in Section 5.

Until recently, the most widely used MCMC procedure was *random walk Metropolis*, which uses a Gaussian random walk proposal $p(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t|\mathbf{x}_{t-1}, \Sigma)$, where Σ is typically diagonal. Modern packages such as PyMC3 allow for automatic tuning of the proposal width Σ, which is critical to achieve good performance. We also consider Cauchy and Laplace distributed proposals.

We include Hamiltonian Monte Carlo (HMC) [Duane et al., 1987] methods, which also utilize gradient information to more efficiently "explore" the space. Recently, the No-U-Turn-Sampler (NUTS) [Hoffman and Gelman, 2014] was introduced as an extension of HMC that automatically adapts some of its tuning parameters in order to attempt high off-the-shelf performance. We include an alternate auxiliary variable method known as *slice sampling* [Neal, 2003], which we apply in a coordinate Gibbs-like fashion.

We also alternate different proposals to form compound proposals. For instance, we consider mixing expensive efficient proposals like NUTS with cheap inefficient proposals like random walk Metropolis.

Finally, we consider an unconventional sampler known as *emcee* [Foreman-Mackey et al., 2013], which is popular in fields such as astrophysics. However, it has not gained much use in machine learning. It works by running multiple "walkers" to explore the space in parallel. Emcee is very fast and can be parallelized, but its efficacy in higher dimensions is somewhat controversial.

Initialization The accuracy of MCMC based estimates are a function of two factors: the *burn-in time* and *the mixing time*. Burn-in time, or time until convergence, is how many steps k are required before $p(\mathbf{x}_k) \approx p^\star$ if $\mathbf{x}_0 \sim p_0$, where p_0 is some distribution to initialize the chain. The mixing time, or memory length, is how long it takes to get an independent sample once a chain has converged: how many steps k are required before $\mathrm{MI}(\mathbf{x}_k; \mathbf{x}_0) \approx 0$ if $\mathbf{x}_0 \sim p^\star$. The burn-in time is crucially dependent on the initialization while the mixing time is purely a function of the proposal.

In order to evaluate these two effects separately, we offer two options for initialization: 1) initialize the chain from an exact sample (because we can do that with the benchmark density examples), or 2) initialize from an ADVI [Kucukelbir et al., 2017] fit to the example density. Additionally, most methods benefit from a prior guess at the relative scale of the variables before tuning. We can use the resulting scales from ADVI for this purpose as well. This allows us to *separate the effects* of initialization and mixing. We use the PyMC3 defaults for these tuning parameters as that is what a practitioner is most likely to use in practice. However, alternate schemes can certainly be used within the benchmark.

4.5 PHASE 4: LOG PERFORMANCE

Each sampler is run for a fixed time limit of 15 minutes of CPU time. We log the performance of the chain along a uniform grid of 100 points in time (i.e., every 9s) to monitor *real* convergence over time. Fair evaluation requires evaluating each sampler with a fixed time budget rather than a fixed number of samples. We expect samplers such as NUTS to be very efficient and high performing on a per-sample basis. However, they require significantly more computation (including gradients) per sample than simpler methods. Therefore, their comparison is not as obvious a-priori. We also log the traditional MCMC diagnostics of each chain.

4.6 PHASE 5: ANALYZE

To summarize the performance of a Markov chain in comparison with ground-truth samples we need to define some evaluation quantities. First, recall that we have K Markov chains $\{\mathbf{x}_{1:N_k}\}_{k=1}^K$ for each example $p^\star \in \mathcal{M}$ and sampler $S \in \mathcal{S}$.

Each sampler is evaluated on each example separately and can be scored relative to a variety of *estimators* $\hat{\theta}(\mathbf{x}_{1:N})$. Analogous to (3), we can score the samples of a Markov chain by the closeness of its mean on a dimension d to the ground-truth samples: $\theta = \mathbb{E}[x_d]$ and $\hat{\theta}(\mathbf{x}_{1:N}) = \frac{1}{N}\sum_{i=1}^N [\mathbf{x}_i]_d$. We can also consider how close the variance of the Markov chain samples match the ground-truth samples: $\theta = \mathrm{Var}[x_d]$. This flexibility is a generalization of ESS. As in (3), we assume the estimators $\hat{\theta}$ are unbiased, and just as with the sample mean $\hat{\mu}$: $\mathrm{Var}_{p^\star}[\hat{\theta}] \propto N^{-1}$. Furthermore, we assume here that each dimension of the samples $\mathbf{x}$ has been standardized using the variance of the ground-truth samples, which makes the estimation errors on each dimension d comparable even when their units differ.

Real ESS In analogy to the ESS diagnostic we define the *real ESS* (RESS) based on the estimation error relative to the ground-truth:

$$\mathrm{RESS} := \frac{R}{\text{mean sq. error}} = \frac{RK}{\sum_{k=1}^K (\hat{\theta}_k - \theta)^2} \in \mathbb{R}^+ \,,$$
$$R := \mathbb{E}_{p^\star}[(\hat{\theta} - \theta)^2] = N\mathrm{Var}_{p^\star}[\hat{\theta}] \in \mathbb{R}^+ \,, \qquad (4)$$

where K is the number of independent MCMC chains and R is a constant to make RESS comparable across different types of estimators $\hat{\theta}$. It also ensures that RESS

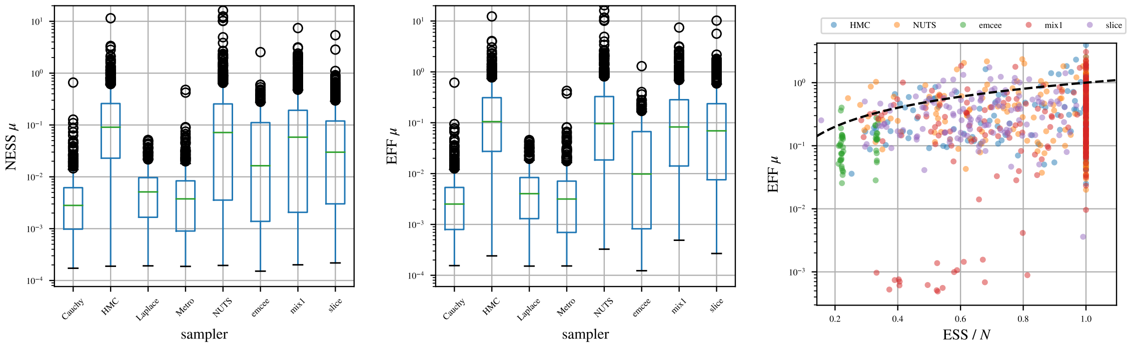

Figure 3: Performance summaries: The box plots demonstrate the distribution on NESS (left) and efficiency (center) conditional on the sampler achieving an RESS of at least 12 to only show the mode where the samplers don't completely fail. We also show a calibration plot to assess if ESS is a good predictor of efficiency with the diagonal in dashed black. Cauchy and Laplace refer to random walk Metropolis with these corresponding proposals.

tends towards N when the samples are iid. We do not need the Var$[x]$ term from (3) because the samples have been standardized using the ground-truth samples' scale. If the estimator θ in (4) is the mean μ_d, then $R = 1$. In this case, the RESS measures the exact same expectation (expected square loss) as ESS attempts to estimate. Therefore, if the chain is sufficiently long for accurate estimation of ESS, the two metrics should converge. For variance σ_d^2 estimation, $R = 2$ in large N.

We also consider the Kolmogorov-Smirnov (KS) distance between the samples and the ground-truth samples as a metric.[1] This also results in a separate metric on each marginal. To match the N^{-1} convergence assumption of (4) we use $\sum_{k=1}^{K} \mathrm{KS}_d(\mathbf{x}_{1:N}^k, p^\star)^2$ as the denominator in (4), where KS_d signifies the KS distance on the marginal x_d. By numerically integrating (4) with the Kolmogorov distribution, one finds that $R = 0.822$.

RESS is also general in that we can sensibly combine the errors across dimensions by evaluating multivariate estimators $\hat{\boldsymbol{\theta}} \in \mathbb{R}^D$:

$$\mathrm{RESS} = \frac{RKD}{\sum_{k=1}^{K} ||\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}||_2^2} \in \mathbb{R}^+ \,. \qquad (5)$$

This assumes that $\hat{\boldsymbol{\theta}}$ is an unbiased estimator of $\boldsymbol{\theta}$. This, like (4), tends towards N for iid samples.

Because $p^\star$ may be complex, yet cheap to take many (e.g., 10^4) iid samples from, we use the ground-truth samples from $p^\star$ to estimate $\boldsymbol{\theta}$ for use in (4). The error in estimating $\boldsymbol{\theta}$ is negligible compared to $\hat{\theta}_k - \theta$. Likewise, for the KS metric we use a two-sample KS distance between the MCMC samples and ground-truth from $p^\star$.

Efficiency Likewise, it is useful for practitioners to get a ballpark estimate of the *efficiency* of a sampler:

$$\mathrm{EFF} := \frac{\mathrm{RESS}}{N} \in \mathbb{R}^+ \,. \qquad (6)$$

If the number of samples per chain N differs across chains, it is more appropriate to use the harmonic mean of N than the mean; this ensures that EFF tends towards unity when samples are drawn iid from $p^\star$. Although EFF is useful, RESS is more appropriate for comparisons between samplers. Thinning can increase EFF without increasing estimation accuracy.

Normalization When looking at the distribution of sampler performance across examples it is more appropriate to look at normalized ESS (NESS):

$$\mathrm{NESS} := \frac{\mathrm{RESS}}{\mathrm{median}_{S \in \mathcal{S}}\, N_S} \in \mathbb{R}^+ \,, \qquad (7)$$

where the median is taken across different samplers on the same example. The RESS, when evaluating with a fixed time limit, varies widely across examples. The computational cost of each sample varies greatly between benchmark examples.

ESS Deviation In order to evaluate the diagnostics in a meta-analysis, we define the ESS deviation (ESSD) metric, which gives a sense on whether the ESS is biased or a generally poor predictor of estimation accuracy. The ESSD is defined as:

$$\mathrm{ESSD} := \Phi^{-1}\left(\chi_K^2 \mathrm{CDF}\left(\frac{\mathrm{ESS}}{\mathrm{RESS}} K\right)\right) \in \mathbb{R} \,, \qquad (8)$$

where $\Phi^{-1}(\cdot)$ is the inverse CDF of the standard normal. ESSD has a standard normal distribution (under CLT assumptions) if the estimates are derived ESS iid samples;

[1] Recall that the KS distance between samples $x_{1:N}$ and a CDF F is given by $\max_a |\hat{F}(a) - F(a)|$ where $\hat{F}$ is the empirical CDF on x.

Table 1: Quantitative summary on sampler performance. We show the NESS on various estimation tasks (e.g., μ vs σ^2) averaged over all examples on the left. The right shows the probability of success, i.e., how often RESS ≥ 12. The first three rows are different proposals for random walk Metropolis. Mix is a compound proposal of NUTS and Gauss. For both NESS and prob. success, higher is better.

	NESS			prob. success		
sampler	KS	μ	σ^2	KS	μ	σ^2
Cauchy	.004	.004	.003	.604	.582	.441
Laplace	.007	.004	.006	.566	.547	.439
Gauss	.007	.005	.007	.585	.565	.436
HMC	.061	.151	.106	.580	.604	.531
NUTS	**.068**	**.375**	**.115**	.875	.783	.711
emcee	.016	.038	.025	.389	.489	.379
mix	.067	.164	.113	**.911**	**.825**	**.715**
slice	.044	.078	.070	.745	.703	.643

ESSD > 0 indicates the estimation is higher error than expected from ESS. More precisely, if $\hat{\theta}$ is derived from m iid samples then,

$$\hat{\theta} \xrightarrow{d} \mathcal{N}(\theta, \sqrt{R/m}) \implies \sqrt{m/R}(\hat{\theta} - \theta) \sim \mathcal{N}(0,1)$$
$$\implies \sum_{k=1}^{K} \frac{m}{R}(\hat{\theta} - \theta)^2 = \frac{m}{\text{RESS}} K \sim \chi_K^2 \,, \qquad (9)$$

which implies that ESSD $\sim \mathcal{N}(0,1)$. Note that (8) is merely a transformation to put the RESS-vs-ESS performance ratio on a standardized scale, which does not cause issues if the central limit theorem (CLT) assumption in (9) does not hold exactly.

Meta-analysis In our meta-analysis, we perform a Gaussian process regression to predict ESSD from ESS $\in \mathbb{R}^+$, Gelman-Rubin GR $\in [1, \infty)$, and Geweke $G \in \mathbb{R}$. We also include the dimension D of the sample space $\mathbf{x}$. Recall that if ESS is a perfect predictor of MCMC performance, then ESSD will resemble white-noise (i.e., iid standard normal). Given that the scales of diagnostics vary widely, we use $\log \text{ESS}$, $\log |\text{GR} - 1|$, and $\log |G|$ to put them all on a sensible scale.

To assess the regression, we test on a held-out 20% test set of unseen examples (i.e., we do random split on a per example basis) to see if we can predict the ESSD on new unseen benchmark examples from the MCMC diagnostics. We compare performance of the regression to linear regression and an iid normal to see if the features provide any predictive gain. Furthermore, we assess the predictive value of each feature by performing the regression after removing each feature and studying the performance delta.

Table 2: Results of meta-analysis. We show the MSE and log-loss of different models attempting to predict the ESSD for mean estimation on a held-out 20% of unseen examples. The log-loss has the advantage that it is parameterization invariant and provides the same results in ESSD or ESS space. The GP- rows show the results of GP regression without the feature named. GP shows the performance of the GP using all features. We assess the statistical significance of the delta to GP using a pairwise t-test in p.

method	MSE	p	NLL (nats)	p
GP	2.8588	–	0	–
GP-D	**2.779(70)**	0.0252	**-0.0096(97)**	0.0504
GP-ESS	3.16(23)	0.0097	0.045(31)	0.0034
GP-G	2.858(1)	0.0198	-0.0001(1)	0.0016
GP-GR	3.17(20)	0.0017	0.045(25)	0.0005
iid	3.30(28)	0.0016	0.067(36)	0.0003
linear	3.03(19)	0.0726	0.027(25)	0.0350

5 RESULTS

We first show an overall summary of final performance using NESS at the end of 15 minutes per chain, with $K = 8$ chains in Table 1. The box plots in Figure 3 provide a sense of the variation. We found the NESS of the samplers to generally be bimodal: either the samples achieve an efficiency above 1%, or they completely fail with an RESS < 1. Therefore, in Figure 3 we show the box plots after excluding the complete failures. Inspired by the rule of $N = 12$ from MacKay [2003], we use an RESS of 12 to threshold failure-vs-success.

Table 1 also provides an overall success probability for each method. Emcee shows the most bimodal performance: while sometimes achieving a high NESS competitive with other advanced methods, it has the lowest success probability. Emcee also has the lowest efficiency of any methods except random walk Metropolis, but emcee makes up for its lack of efficiency with higher per sample speed.

Other results from Figure 3 are unsurprising: NUTS and HMC are the highest performers, despite their higher per sample cost. Slice sampling also makes a "strong showing" with its performance more competitive in the lower dimensional examples. Random walk Metropolis methods generally have an efficiency in the 0.1% to 1% range, while slice sampling and HMC based methods have efficiencies in the ballpark of 2% to 40%, with NUTS showing the highest performance. Emcee seems to vary widely. Note that although the compound proposal (mix) does not substantially increase NESS (over NUTS), when the methods succeed Figure 3, mix increases the chance of success (Table 1).

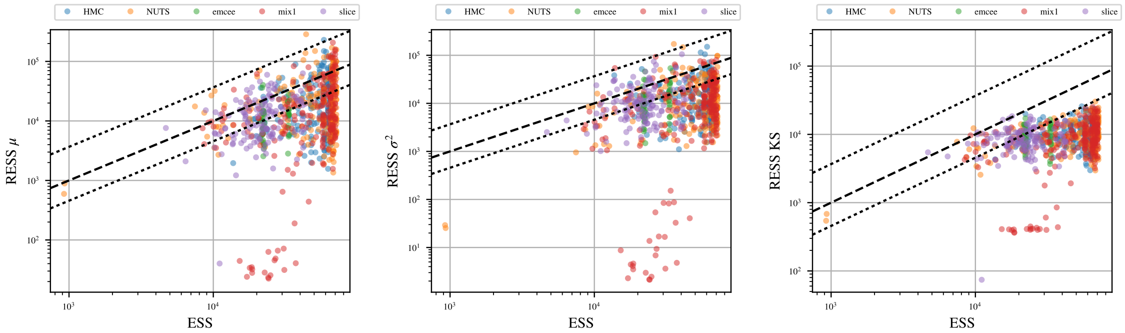

Figure 4: Calibration plots of the ESS diagnostic against real ESS with $\hat{\theta}$ being the mean (left), variance (center), or KS (right). We show the diagonal for a perfect match in dashed black. In dotted black we show the 95% region for what the observed real ESS would be if the estimates $\hat{\theta}$ were derived from ESS iid samples. The RESS is below the lower error bar 55% of the time for mean estimation, 68% for variance, and 83% for KS; these would be 2.5% if a chain with ESS $= m$ were functionally equivalent to m iid samples.

We show calibration plots of ESS in Figure 4 and efficiency in Figure 3. The ESS diagnostic is clearly best calibrated for mean estimation, which is not surprising given it was derived for that purpose. However, the ESS diagnostic clearly has an optimistic bias. These results provide caution of ESS.

Finally, we present the results of the meta-analysis to predict ESS deviation. We report the predictive value provided by various features in Table 2 by showing how much performance changes when they are removed. ESS appears very predictive in Figure 4, but the relationship has already largely been accounted for with ESSD (8). In log-loss, the remaining predictive utility of ESS equals that of Gelman-Rubin. Geweke and the dimension D show no predictive utility. Predictive performance of ESSD goes up when they are removed, which indicates they are of little utility when assessing the validity of a Markov chain. One expects sampling to be more difficult in higher dimensions D, however this slower mixing may already be evident from ESS and Gelman-Rubin.

6 CONCLUSIONS

We have presented a general system to benchmark the real performance of MCMC samplers on realistic problems. The data-driven nature of the benchmark makes it a highly novel development. This benchmark is intended to become a general service that will become as widespread as COCO or MLcomp. Careful attention has been paid to fairly and sensibly derive metrics that compare samplers. This benchmark will evolve with time by including ever more models in phase 1 and more advanced example densities in phase 2. New and more sophisticated samplers can easily be added in phase 3.

References

B. Ballnus, S. Hug, K. Hatz, L. Görlitz, J. Hasenauer, and F. J. Theis. Comprehensive benchmarking of Markov chain Monte Carlo methods for dynamical systems. *BMC Systems Biology*, 11(1):63, 2017.

B. Carpenter, A. Gelman, M. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. A. Brubaker, J. Guo, P. Li, and A. Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 20: 1–37, 2016.

M. K. Cowles and B. P. Carlin. Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91 (434):883–904, 1996.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society (Series B)*, pages 1–38, 1977.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv:1605.08803*, 2016.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. Emcee: The MCMC hammer. *Publications of the Astronomical Society of the Pacific*, 125(925):306, 2013.

A. Gelman and D. B. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, pages 457–472, 1992.

J. Geweke. Evaluating the accuracy of sampling-based approaches to calculating posterior moments. *Bayesian Statistics 4*, 1992.

I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macia, B. Ray, M. Saeed, A. Statnikov, et al. Design of the 2015 ChaLearn AutoML challenge. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

N. Hansen, A. Auger, O. Mersmann, T. Tusar, and D. Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *arXiv:1603.08785*, 2016.

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

W. H. Jefferys and J. O. Berger. Ockham's razor and Bayesian analysis. *American Scientist*, 80(1):64–72, 1992.

I. T. Jolliffe. Principal component analysis and factor analysis. In *Principal Component Analysis*, pages 115–128. Springer, 1986.

R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal. Markov chain Monte Carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100, 1998.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18 (14):1–45, 2017.

J. R. Lloyd, D. K. Duvenaud, R. B. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence*, pages 1242–1250, 2014.

D. J. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

R. M. Neal. Slice sampling. *Annals of Statistics*, pages 705–741, 2003.

C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

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

K. Sharp and M. Rattray. Dense message passing for sparse principal component analysis. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 725–732, 2010.

M. B. Thompson. Introduction to SamplerCompare. *Journal of Statistical Software*, 43(12):1–10, 2011.

B. Uria, I. Murray, and H. Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, pages 2175–2183, 2013.