Benchmarking Simulation-Based Inference

Jan-Matthis Lueckmann[1,2] **Jan Boelts**[2] **David S. Greenberg**[2,3]
Pedro J. Gonçalves[4] **Jakob H. Macke**[1,2,5]

[1]University of Tübingen [2]Technical University of Munich [3]Helmholtz Centre Geesthacht
[4]Research Center caesar [5]Max Planck Institute for Intelligent Systems, Tübingen

Abstract

Recent advances in probabilistic modelling have led to a large number of simulation-based inference algorithms which do not require numerical evaluation of likelihoods. However, a public benchmark with appropriate performance metrics for such 'likelihood-free' algorithms has been lacking. This has made it difficult to compare algorithms and identify their strengths and weaknesses. We set out to fill this gap: We provide a benchmark with inference tasks and suitable performance metrics, with an initial selection of algorithms including recent approaches employing neural networks and classical Approximate Bayesian Computation methods. We found that the choice of performance metric is critical, that even state-of-the-art algorithms have substantial room for improvement, and that sequential estimation improves sample efficiency. Neural network-based approaches generally exhibit better performance, but there is no uniformly best algorithm. We provide practical advice and highlight the potential of the benchmark to diagnose problems and improve algorithms. The results can be explored interactively on a companion website. All code is open source, making it possible to contribute further benchmark tasks and inference algorithms.

1 Introduction

Many domains of science, engineering, and economics make extensive use of models implemented as stochastic numerical simulators (Gourieroux et al., 1993; Ratmann et al., 2007; Alsing et al., 2018; Brehmer et al., 2018; Karabatsos and Leisen, 2018; Gonçalves et al., 2020). A key challenge when studying and validating such simulation-based models is the statistical identification of parameters which are consistent with observed data. In many cases, calculation of the likelihood is intractable or impractical, rendering conventional approaches inapplicable. The goal of simulation-based inference (SBI), also known as 'likelihood-free inference', is to perform Bayesian inference without requiring numerical evaluation of the likelihood function (Sisson et al., 2018; Cranmer et al., 2020). In SBI, it is generally not required that the simulator is differentiable, nor that one has access to its internal random variables.

In recent years, several new SBI algorithms have been developed (e.g., Gutmann and Corander, 2016; Papamakarios and Murray, 2016; Lueckmann et al., 2017; Chan et al., 2018; Greenberg et al., 2019; Papamakarios et al., 2019b; Prangle, 2019; Brehmer et al., 2020; Hermans et al., 2020; Järvenpää et al., 2020; Picchini et al., 2020; Rodrigues et al., 2020; Thomas et al., 2020), energized, in part, by advances in probabilistic machine learning (Rezende and Mohamed, 2015; Papamakarios et al., 2017, 2019a). Despite—or possibly *because*—of these rapid and exciting developments, it is currently difficult to assess how different approaches relate to each other theoretically and empirically: First, different studies often use different tasks and metrics for comparison, and comprehensive comparisons on multiple tasks and simulation budgets are rare. Second, some commonly employed metrics might not be appropriate or might be biased through the choice of hyperparameters. Third, the absence of a benchmark has made it necessary to reimplement tasks and algorithms for each new study. This practice is wasteful, and makes it hard to rapidly evaluate the potential of new algorithms. Overall, it is difficult to discern the most promising approaches and decide on which algorithm to use when. These problems are exacerbated by the interdisciplinary nature of research on SBI, which has

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

led to independent development and co-existence of closely-related algorithms in different disciplines.

There are many exciting challenges and opportunities ahead, such as the scaling of these algorithms to high-dimensional data, active selection of simulations, and gray-box settings, as outlined in Cranmer et al. (2020). To tackle such challenges, researchers will need an extensible framework to compare existing algorithms and test novel ideas. Carefully curated, a benchmark framework will make it easier for researchers to enter SBI research, and will fuel the development of new algorithms through community involvement, exchange of expertise and collaboration. Furthermore, benchmarking results could help practitioners to decide which algorithm to use on a given problem of interest, and thereby contribute to the dissemination of SBI.

The catalyzing effect of benchmarks has been evident, e.g., in computer vision (Russakovsky et al., 2015), speech recognition (Hirsch and Pearce, 2000; Wang et al., 2018), reinforcement learning (Bellemare et al., 2013; Duan et al., 2016), Bayesian deep learning (Filos et al., 2019; Wenzel et al., 2020), and many other fields drawing on machine learning. Open benchmarks can be an important component of transparent and reproducible computational research. Surprisingly, a benchmark framework for SBI has been lacking, possibly due to the challenging endeavor of designing benchmarking tasks and defining suitable performance metrics.

Here, we begin to address this challenge, and provide a benchmark framework for SBI to allow rapid and transparent comparisons of current and future SBI algorithms: First, we selected a set of initial algorithms representing distinct approaches to SBI (Fig. 1; Cranmer et al., 2020). Second, we analyzed multiple performance metrics which have been used in the SBI literature. Third, we implemented ten tasks including tasks popular in the field. The shortcomings of commonly used metrics led us to focus on tasks for which a likelihood *can* be evaluated, which allowed us to calculate reference ('ground-truth') posteriors. These reference posteriors are made available to allow rapid evaluation of SBI algorithms. Code for the framework is available at `github.com/sbi-benchmark/sbibm` and we maintain an interactive version of all results at `sbi-benchmark.github.io`.

The full potential of the benchmark will be realized when it is populated with additional community-contributed algorithms and tasks. However, our initial version already provides useful insights: 1) the choice of performance metric is critical; 2) the performance of the algorithms on some tasks leaves substantial room for improvement; 3) sequential estimation generally improves sample efficiency; 4) for small and moderate simulation budgets, neural-network based approaches outperform classical ABC algorithms, confirming recent progress in the field; and 5) there is no algorithm to rule them all. The performance ranking of algorithms is task-dependent, pointing to a need for better guidance or automated procedures for choosing which algorithm to use when. We highlight examples of how the benchmark can be used to diagnose shortcomings of algorithms and facilitate improvements. We end with a discussion of the limitations of the benchmark.

2 Benchmark

The benchmark consists of a set of algorithms, performance metrics and tasks. Given a prior $p(\boldsymbol{\theta})$ over parameters $\boldsymbol{\theta}$, a simulator to sample $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$ and an observation $\mathbf{x}_o$, an algorithm returns an approximate posterior $q(\boldsymbol{\theta}|\mathbf{x}_o)$, or samples from it, $\boldsymbol{\theta} \sim q$. The approximate solution is tested, according to a performance metric, against a reference posterior $p(\boldsymbol{\theta}|\mathbf{x}_o)$.

2.1 Algorithms

Following the classification introduced in the review by Cranmer et al. (2020), we selected algorithms addressing SBI in four distinct ways, as schematically depicted in Fig. 1. An important difference between algorithms is how new simulations are acquired: Sequential algorithms adaptively choose informative simulations to increase sample efficiency. While crucial for expensive simulators, it can require non-trivial algorithmic steps and hyperparameter choices. To evaluate whether the potential is realized empirically and justifies the algorithmic burden, we included sequential and non-sequential counterparts for algorithms of each category.

Keeping our initial selection focused allowed us to carefully consider implementation details and hyperparameters: We extensively explored performance and sensitivity to different choices in more than 10k runs, all results and details of which can be found in Appendix H. Our selection is briefly described below, full algorithm details are in Appendix A.

REJ-ABC and SMC-ABC. Approximate Bayesian Computation (ABC, Sisson et al., 2018) is centered around the idea of Monte Carlo rejection sampling (Tavaré et al., 1997; Pritchard et al., 1999). Parameters $\boldsymbol{\theta}$ are sampled from a proposal distribution, simulation outcomes $\mathbf{x}$ are compared with observed data $\mathbf{x}_o$, and are accepted or rejected depending on a (user-specified) distance function and rejection criterion. While rejection ABC (REJ-ABC) uses the prior as a proposal distribution, the efficiency can be improved by using sequentially refined proposal distributions (SMC-ABC, Beaumont et al., 2002; Marjoram and Tavaré, 2006;

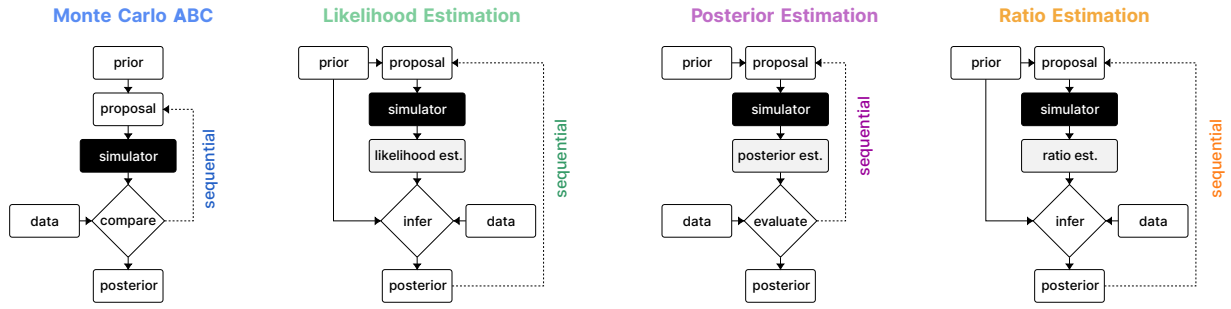

Figure 1: **Overview of algorithms.** We compare algorithms belonging to four distinct approaches to SBI: Classical ABC approaches as well as model-based approaches approximating likelihoods, posteriors, or density ratios. We contrast algorithms that use the prior distribution to propose parameters against ones that sequentially adapt the proposal. Classification and schemes following Cranmer et al. (2020).

Sisson et al., 2007; Toni et al., 2009; Beaumont et al., 2009). We implemented REJ-ABC with quantile-based rejection and used the scheme of Beaumont et al. (2009) for SMC-ABC. We extensively varied hyperparameters and compared the implementation of an ABC-toolbox (Klinger et al., 2018) against our own (Appendix H). We investigated linear regression adjustment (Blum and François, 2010) and the summary statistics approach by Prangle et al. (2014) (Suppl. Fig. 1).

NLE and SNLE. Likelihood estimation (or 'synthetic likelihood') algorithms learn an approximation to the intractable likelihood, for an overview see Drovandi et al. (2018). While early incarnations focused on Gaussian approximations (SL; Wood, 2010), recent versions utilize deep neural networks (Papamakarios et al., 2019b; Lueckmann et al., 2019) to approximate a density over $\mathbf{x}$, followed by MCMC to obtain posterior samples. Since we primarily focused on these latter versions, we refer to them as neural likelihood estimation (NLE) algorithms, and denote the sequential variant with proposals as SNLE. In particular, we used the scheme proposed by Papamakarios et al. (2019b) which uses masked autoregressive flows (MAFs, Papamakarios et al., 2017) for density estimation. We improved MCMC sampling for (S)NLE and compared MAFs against Neural Spline Flows (NSFs; Durkan et al., 2019), see Appendix H.

NPE and SNPE. Instead of approximating the likelihood, these approaches directly target the posterior. Their origins date back to regression adjustment approaches (Blum and François, 2010). Modern variants (Papamakarios and Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019) use neural networks for density estimation (approximating a density over $\boldsymbol{\theta}$). Here, we used the recent algorithmic approach proposed by Greenberg et al. (2019) for sequential acquisitions. We report performance using NSFs for density estimation, which outperformed MAFs (Appendix H).

NRE and SNRE. Ratio Estimation approaches to SBI use classifiers to approximate density ratios (Izbicki et al., 2014; Pham et al., 2014; Cranmer et al., 2015; Dutta et al., 2016; Durkan et al., 2020; Thomas et al., 2020). Here, we used the recent approach proposed by Hermans et al. (2020) as implemented in Durkan et al. (2020): A neural network-based classifier approximates probability ratios and MCMC is used to obtain samples from the posterior. SNRE denotes the sequential variant of neural ratio estimation (NRE). In Appendix H we compare different classifier architectures for (S)NRE.

In addition, we benchmarked Random Forest ABC (RF-ABC; Raynal et al., 2019), a recent ABC variant, and Synthetic Likelihood (SL; Wood, 2010), mentioned above. However, RF-ABC only targets individual parameters (i.e. assumes posteriors to factorize), and SL requires new simulations for every MCMC step, thus requiring orders of magnitude more simulations than other algorithms. Therefore, we report results for these algorithms separately, in Suppl. Fig. 2 and Suppl. Fig. 3, respectively.

Algorithms can be grouped with respect to how their output is represented: 1) some return samples from the posterior, $\boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\mathbf{x}_o)$ (REJ-ABC, SMC-ABC); 2) others return samples and allow evaluation of unnormalized posteriors $\tilde{q}(\boldsymbol{\theta}|\mathbf{x}_o)$ ((S)NLE, (S)NRE); and 3) for some, the posterior density $q(\boldsymbol{\theta}|\mathbf{x}_o)$ can be evaluated and sampled directly, without MCMC ((S)NPE). As discussed below, these properties constrain the metrics that can be used for comparison.

2.2 Performance metrics

Choice of a suitable performance metric is central to any benchmark. As the goal of SBI algorithms is to perform full inference, the 'gold standard' would be to quantify the similarity between the true posterior and the inferred one with a suitable distance (or di-

vergence) measure on probability distributions. This would require both access to the ground-truth posterior, and a reliable means of estimating similarity between (potentially) richly structured distributions. Several performance metrics have been used in past research, depending on the constraints imposed by knowledge about ground-truth and the inference algorithm (see Table 1). In real-world applications, typically only the observation $\mathbf{x}_o$ is known. However, in a benchmarking setting, it is reasonable to assume that one has at least access to the ground-truth parameters $\boldsymbol{\theta}_o$. There are two commonly used metrics which only require $\boldsymbol{\theta}_o$ and $\mathbf{x}_o$, but suffer severe drawbacks for our purposes:

Probability $\boldsymbol{\theta}_o$. The negative log probability of true parameters averaged over different $(\boldsymbol{\theta}_o, \mathbf{x}_o)$, $-\mathbb{E}[\log q(\boldsymbol{\theta}_o|\mathbf{x}_o)]$, has been used extensively in the literature (Papamakarios and Murray, 2016; Durkan et al., 2018; Greenberg et al., 2019; Papamakarios et al., 2019b; Durkan et al., 2020; Hermans et al., 2020). Its appeal lies in the fact that one does not need access to the ground-truth posterior. However, using it only for a small set of $(\boldsymbol{\theta}_o, \mathbf{x}_o)$ is highly problematic: It is only a valid performance measure if averaged over a large set of observations sampled from the prior (Talts et al., 2018, detailed discussion including connection to simulation-based calibration in Appendix M). For reliable results, one would require inference for hundreds of $\mathbf{x}_o$ which is only feasible if inference is rapid (amortized) and the density q can be evaluated directly (among the algorithms used here this applies only to NPE).

Posterior-Predictive Checks (PPCs). As the name implies, PPCs should be considered a mere check rather than a metric, although the *median distance* between predictive samples and $\mathbf{x}_o$ has been reported in the SBI literature (Papamakarios et al., 2019b; Greenberg et al., 2019; Durkan et al., 2020). A failure mode of such a metric is that an algorithm obtaining a good MAP point estimate, could perfectly pass this check even if the estimated posterior is poor. Empirically, we found median-distances (MEDDIST) to be in disagreement with other metrics (see Results).

The shortcomings of these commonly-used metrics led us to focus on tasks for which it is possible to get samples from ground-truth posterior $\boldsymbol{\theta} \sim p$, thus allowing us to use metrics based on two-sample tests:

Maximum Mean Discrepancy (MMD). MMD (Gretton et al., 2012; Sutherland et al., 2017) is a kernel-based 2-sample test. Recent papers (Papamakarios et al., 2019b; Greenberg et al., 2019; Hermans et al., 2020) reported MMD using translation-invariant Gaussian kernels with length scales determined by the median heuristic (Ramdas et al., 2015). We empirically found that MMD can be sensitive to hyperparameter choices, in particular on posteriors with multiple modes and length scales (see Results and Liu et al., 2020).

Classifier 2-Sample Tests (C2ST). C2STs (Friedman, 2004; Lopez-Paz and Oquab, 2017) train a classifier to discriminate samples from the true and inferred posteriors, which makes them simple to apply and easy to interpret. Therefore, we prefer to report and compare algorithms in terms of accuracy in classification-based tests. In the context of SBI, C2ST has e.g. been used in Gutmann et al. (2018); Dalmasso et al. (2020).

Other metrics that could be used include:

Kernelized Stein Discrepancy (KSD). KSD (Liu et al., 2016; Chwialkowski et al., 2016) is a 1-sample test, which require access to $\nabla_{\boldsymbol{\theta}} \tilde{p}(\boldsymbol{\theta}|\mathbf{x}_o)$ rather than samples from p ($\tilde{p}$ is the unnormalized posterior). Like MMD, current estimators use translation-invariant kernels.

f-Divergences. Divergences such as Total Variation (TV) divergence and KL divergences can only be computed when the densities of true and approximate posteriors can be evaluated (Table 1). Thus, we did not use f-divergences for the benchmark.

Full discussion and details of metrics in Appendix M.

2.3 Tasks

The preceding considerations guided our selection of inference tasks: We focused on tasks for which reference posterior samples $\boldsymbol{\theta} \sim p$ can be obtained, to allow calculation of 2-sample tests. We focused on eight purely statistical problems and two problems relevant in applied domains, with diverse dimensionalities of parameters and data (details in Appendix T):

Gaussian Linear/Gaussian Linear Uniform. We included two versions of simple, linear, 10-d Gaussian models, in which the parameter $\boldsymbol{\theta}$ is the mean, and the covariance is fixed. The first version has a Gaussian (conjugate) prior, the second one a uniform prior. These tasks allow us to test how algorithms deal with trivial scaling of dimensionality, as well as truncated support.

SLCP/SLCP Distractors. A challenging inference task designed to have a simple likelihood and a complex posterior (Papamakarios et al., 2019b; Greenberg et al., 2019): The prior is uniform over five parameters $\boldsymbol{\theta}$ and the data are a set of four two-dimensional points sampled from a Gaussian likelihood whose mean and variance are nonlinear functions of $\boldsymbol{\theta}$. This induces a complex posterior with four symmetrical modes and vertical cut-offs. We included a second version with 92 additional, non-informative outputs (distractors) to test the ability to detect informative features.

Bernoulli GLM/Bernoulli GLM Raw. 10-parameter Generalized Linear Model (GLM) with

Table 1: **Applicability of metrics given knowledge about ground truth and algorithm.** Whether a metric can be used depends on *both* what is known about the ground-truth of an inference task and what an algorithm returns: Information about ground truth can vary between just having observed data $\mathbf{x}_o$ (typical setting in practice), knowing the generating parameter $\boldsymbol{\theta}_o$, having posterior samples, gradients, or being able to evaluate the true posterior p. Tilde denotes unnormalized distributions. Access to information is cumulative.

	Ground truth $\rightarrow$				
$\downarrow$ **Algorithm**	$\mathbf{x}_o$	$\boldsymbol{\theta}_o$	$\boldsymbol{\theta} \sim p$	$\nabla \tilde{p}(\boldsymbol{\theta}\|\mathbf{x}_o)$	$p(\boldsymbol{\theta}\|\mathbf{x}_o)$
$\boldsymbol{\theta} \sim q$	1	1	1, 3	1, 3, 4	1, 3, 4
$\tilde{q}(\boldsymbol{\theta}\|\mathbf{x}_o)$	1	1	1, 3	1, 3, 4	1, 3, 4
$q(\boldsymbol{\theta}\|\mathbf{x}_o)$	1	1, 2	1, 2, 3	1, 2, 3, 4	1, 2, 3, 4, 5

1 = PPCs, 2 = Probability $\boldsymbol{\theta}_0$, 3 = 2-sample tests, 4 = 1-sample tests, 5 = f-divergences.

Bernoulli observations. Inference was either performed on sufficient statistics (10-d) or raw data (100-d).

Gaussian Mixture. This inference task, introduced by Sisson et al. (2007), has become common in the ABC literature (Beaumont et al., 2009; Toni et al., 2009; Simola et al., 2020). It consists of a mixture of two two-dimensional Gaussian distributions, one with much broader covariance than the other.

Two Moons. A two-dimensional task with a posterior that exhibits both global (bimodality) and local (crescent shape) structure (Greenberg et al., 2019) to illustrate how algorithms deal with multimodality.

SIR. Dynamical systems represent paradigmatic use cases for SBI. SIR is an influential epidemiological model describing the dynamics of the number of individuals in three possible states: susceptible S, infectious I, and recovered or deceased, R. We infer the contact rate β and the mean recovery rate γ, given observed infection counts I at 10 evenly-spaced time points.

Lotka-Volterra. An influential model in ecology describing the dynamics of two interacting species, widely used in SBI studies. We infer four parameters $\boldsymbol{\theta}$ related to species interaction, given the number of individuals in both populations at 10 evenly-spaced points in time.

2.4 Experimental Setup

For each task, we sampled 10 sets of true parameters from the prior and generated corresponding observations $(\boldsymbol{\theta}_o, \mathbf{x}_o)_{1:10}$. For each observation, we generated 10k samples from the reference posterior. Some reference posteriors required a customised (likelihood-based) approach (Appendix B).

In SBI, it is typically assumed that total computation cost is dominated by simulation time. We therefore report performance at different simulation budgets. For each observation, each algorithm was run with a simulation budget ranging from 1k to 100k simulations.

For each run, we calculated metrics described above. To estimate C2ST accuracy, we trained a multilayer perceptron to tell apart approximate and reference posterior samples and performed five-fold cross-validation. We used two hidden layers, each with 10 times as many ReLu units as the dimensionality of the data. We also measured and report runtimes (Appendix R).

2.5 Software

Code. All code is released publicly at `github.com/sbi-benchmark/sbibm`. Our framework includes tasks, reference posteriors, metrics, plotting, and infrastructure tooling and is designed to be 1) easily extensible, 2) used with external toolboxes implementing algorithms. All tasks are implemented as probabilistic programs in `Pyro` (Bingham et al., 2019), so that likelihoods and gradients for reference posteriors can be extracted automatically. To make this possible for tasks that use ODEs, we developed a new interface between `DifferentialEquations.jl` (Rackauckas and Nie, 2017; Bezanson et al., 2017) and `PyTorch` (Paszke et al., 2019). In addition, specifying simulators in a probabilistic programming language has the advantage that 'gray-box' algorithms (Brehmer et al., 2020; Cranmer et al., 2020) can be added in the future. We here evaluated algorithms implemented in `pyABC` (Klinger et al., 2018), `pyabcranger` (Collin et al., 2020), and `sbi` (Tejero-Cantero et al., 2020). See Appendix B for details and existing SBI toolboxes.

Reproducibility. Instructions to reproduce experiments on cloud-based infrastructure are in Appendix B.

Website. Along with the code, we provide a web interface which allows interactive exploration of all the results (`sbi-benchmark.github.io`; Appendix W).

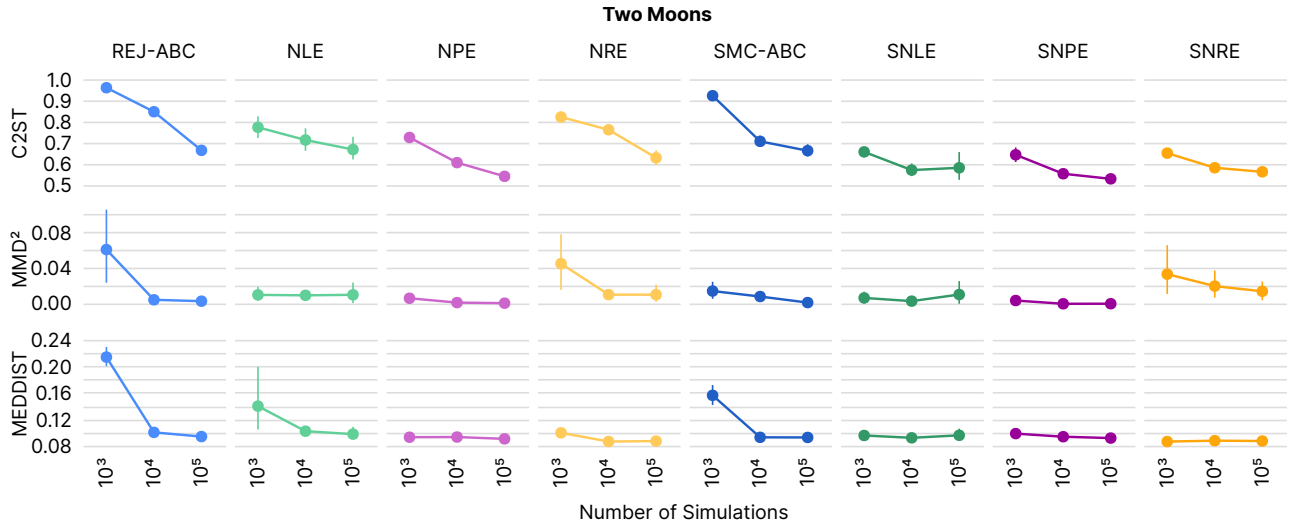

Figure 2: **Performance on Two Moons according to various metrics.** Best possible performance would be 0.5 for C2ST, 0 for MMD^2 and MEDDIST. Results for 10 observations each, means and 95% confidence intervals.

3 Results

We first consider empirical results on a single task, Two Moons, according to different metrics, which illustrate the following important insight:

#1: Choice of performance metric is key. While C2ST results on Two Moons show that performance increases with higher simulation budgets and that sequential algorithms outperform non-sequential ones for low to medium budgets, these results were not reflected in MMD and MEDDIST (Fig. 2): In our analyses, we found MMD to be sensitive to hyperparameter choices, in particular on tasks with complex posterior structure. When using the commonly employed median heuristic to set the kernel length scale on a task with multi-modal posteriors (like Two Moons), MMD had difficulty discerning markedly different posteriors. This can be 'fixed' by using hyperparameters adapted to the task (Suppl. Fig. 4). As discussed above, the median distance (though commonly used) can be 'gamed' by a good point estimate even if the estimated posterior is poor and is thus not a suitable performance metric. Computation of KSD showed numerical problems on Two Moons, due to the gradient calculation.

We assessed relationships between metrics empirically via the correlations across tasks (Suppl. Fig. 5). As discussed above, the log-probability of ground-truth parameters can be problematic when averaged over too few observations (e.g., 10, as is common in the literature): indeed, this metric had a correlation of only 0.3 with C2ST on Two Moons and 0.6 on the SLCP task. Based on these considerations, we used C2ST for reporting performance (Fig. 3; results for MMD, KSD and median distance on the website).

Based on the comparison of the performance across all tasks, we highlight the following main points:

#2: These are not solved problems. C2ST uses an interpretable scale (1 to 0.5), which makes it possible to conclude that, for several tasks, no algorithm could solve them with the specified budget (e.g., SLCP, Lotka-Volterra). This highlights that our problems—though conceptually simple—are challenging, and there is room for development of more powerful algorithms.

#3: Sequential estimation improves sample efficiency. Our results show that sequential algorithms outperform non-sequential ones (Fig. 3). The difference was small on simple tasks (i.e. linear Gaussian cases), yet pronounced on most others. However, we also found these methods to exhibit diminishing returns as the simulation budget grows, which points to an opportunity for future improvements.

#4: Density or ratio estimation-based algorithms generally outperform classical techniques. REJ-ABC and SMC-ABC were generally outperformed by more recent techniques which use neural networks for density- or ratio-estimation, and which can therefore efficiently interpolate between different simulations (Fig. 3). Without such model-based interpolation, even a simple 10-d Gaussian task can be challenging. However, classical rejection-based methods have a computational footprint that is orders of magnitudes smaller, as no network training is involved (Appendix R). Thus, on low-dimensional problems and for cheap simulators, these methods can still be competitive. See Suppl. Fig. 1 for results with additional ABC variants (Blum and François, 2010; Prangle et al., 2014) and Suppl. Fig. 2 for results on RF-ABC.

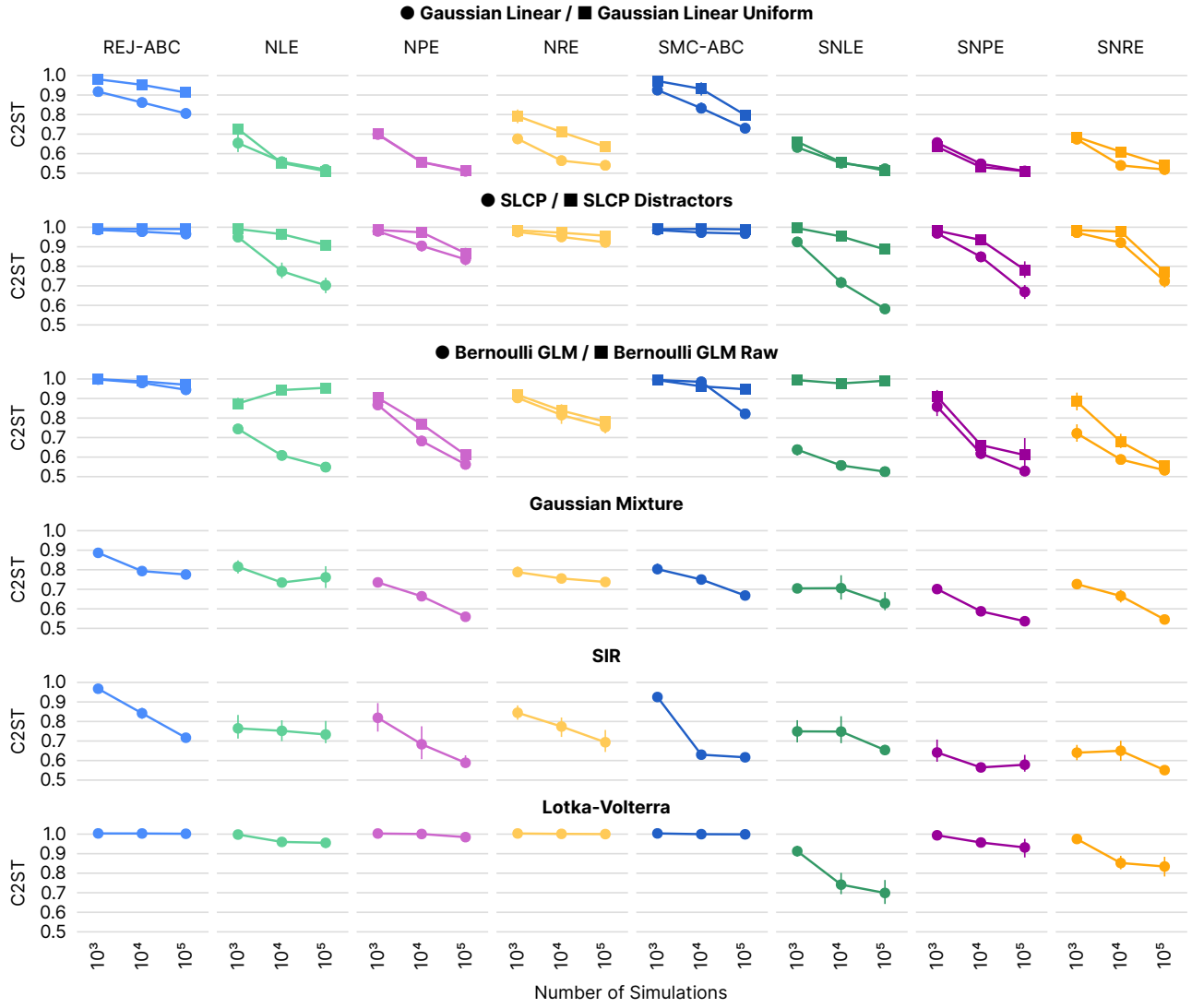

Figure 3: **Performance on other benchmark tasks.** Classification accuracy (C2ST) of REJ-ABC, SMC-ABC, NLE, SNLE, NPE, SNPE, NRE, SNRE for 10 observations each, means and 95% confidence intervals.

#5: No one algorithm to rule them all. Although sequential density or ratio estimation-based algorithms performed better than their non-sequential counterparts, there was no clear-cut answer as to which sequential method (SNLE, SNRE, and SNPE) should be preferred. To some degree, this is to be expected: these algorithms have distinct strengths that can play out differently depending on the problem structure (see discussions e.g., in Greenberg et al., 2019; Durkan et al., 2018, 2020). However, this has not been shown systematically before. We formulate some practical guidelines for choosing appropriate algorithms in Box 1.

#6: The benchmark can be used to diagnose implementation issues and improve algorithms. For example, (S)NLE and (S)NRE rely on MCMC sampling to compute posteriors, and this sampling step can limit the performance. Access to a reference posterior can help identify and improve such issues: We found that single chains initialized by sampling from the prior with axis-aligned slice sampling (as introduced in Papamakarios et al., 2019b) frequently got stuck in single modes. Based on this observation, we changed the MCMC strategy (details in Appendix A), which, though simple, yielded significant performance and speed improvements on the benchmark tasks. Similarly, (S)NLE and (S)NRE improved by transforming parameters to be unbounded: Without transformations, runs on some tasks can get stuck during MCMC sampling (e.g., Lotka-Volterra). While this is common advice for MCMC (Hogg and Foreman-Mackey, 2018), it has been lacking in code and papers of SBI approaches.

We used the benchmark to systematically compare hyperparameters: For example, as density estimators

Do we need the Bayesian posterior, or is a point estimate sufficient?

Our focus was on SBI algorithms that target the Bayesian posterior. If one only aims for a single estimate, optimization methods (e.g. Rios and Sahinidis, 2013; Shahriari et al., 2015) might be more efficient.

Is the simulator really 'black-box'?

The SBI algorithms presented in the benchmark can be applied to any 'black-box' simulator. However, if the likelihood is available, methods exploiting it (e.g. MCMC, variational inference) will generally be more efficient. Similarly, if one has access to the internal random numbers, probabilistic programming approaches (Le et al., 2017; Baydin et al., 2019; Wood et al., 2020) might be preferable. If additional quantities that characterize the latent process are available, i.e., the simulator is 'gray-box', they can be used to augment training data and improve inference (Brehmer et al., 2020; Cranmer et al., 2020).

What domain knowledge do we have about the problem?

For any practical application of SBI, it is worth thinking carefully about domain knowledge. First, knowledge about plausible parameters should inform the choice of the prior. Second, domain knowledge can help design appropriate distance functions or summary statistics required for classical ABC algorithms. When using model-based approaches, domain knowledge can potentially be built into the SBI algorithm itself, for example, by incorporating neural network layers with appropriate inductive biases or invariances.

Do we have, or can we learn summary statistics?

Summary statistics are especially important when facing problems with high-dimensional data: It is important to point out that the posterior given summary statistics $p(\boldsymbol{\theta}|s(\mathbf{x}_o))$ is only equivalent to $p(\boldsymbol{\theta}|\mathbf{x}_o)$ if the summary statistics are sufficient. The problem at hand can guide the manual design of summary statistics that are regarded particularly important or informative. Alternatively, many automatic approaches exist (e.g., Prangle et al., 2014; Charnock et al., 2018; Dinev and Gutmann, 2018) and this is an active area of research (e.g., Chen et al. 2021 recently proposed an approach to learn approximately sufficient statistics for SMC-ABC and (S)NLE). (S)NPE and (S)NRE can directly reduce high-dimensional data as part of their network architectures.

Do we have low-dimensional data and parameters, and a cheap simulator?

If both the parameters and the data (or suitable summary-statistics thereof) are low-dimensional, and a very large number of simulations can be generated, model-free algorithms such as classical ABC can be competitive. These have the benefit of adding little computational overhead. Conversely, for limited simulation budgets and/or higher dimensionalities, approaches that train a model of the likelihood, posterior, or likelihood ratio will generally be preferable.

Are simulations expensive? Can we simulate online?

For time-intensive and complex simulators, it can be beneficial to use *sequential* methods to increase sample efficiency: We found that sequential schemes generally outperformed non-sequential ones. While we focused on simple strategies which use the previous estimate of the posterior to propose new parameters, more sophisticated schemes (e.g., Gutmann and Corander, 2016; Lueckmann et al., 2019; Järvenpää et al., 2019) may increase sample efficiency if only few simulations can be obtained. For some applications, inference is performed on a fixed dataset, and one cannot resort to sequential algorithms.

Do we want to carry out inference once, or repeatedly?

To perform SBI *separately* for different data points (i.e. compute $p(\boldsymbol{\theta}|\mathbf{x}_1), p(\boldsymbol{\theta}|\mathbf{x}_2), \ldots$), methods that allow 'amortization' (NPE) are likely preferable. While NLE and NRE allow amortisation of the neural network, MCMC sampling is required, which takes additional time. Conversely, if we want to run SBI conditioned on many i.i.d. data (e.g. $p(\boldsymbol{\theta}|\mathbf{x}_1, \mathbf{x}_2, \ldots)$) methods based on likelihood or ratio estimation (NLE, NRE), or NPE with exchangeable neural networks (Chan et al., 2018) would be appropriate.

Box 1: **Practitioners' advice for applying SBI algorithms.** Based on our current results and understanding, we provide advice to practitioners seeking to apply SBI. There is no one-fits-all solution—which algorithm to use in practice will depend on the problem at hand. For additional advice, see Cranmer et al. (2020).

for (S)NLE and (S)NPE, we used NSFs (Durkan et al., 2020) which were developed after these algorithms were published. This revealed that higher capacity density estimators were beneficial for posterior but not likelihood estimation (detailed analysis in Appendix H).

These examples show how the benchmark makes it possible to diagnose problems and improve algorithms.

4 Limitations

Our benchmark, in its current form, has several limitations. First, the algorithms considered here do not cover the entire spectrum of SBI algorithms: We did not include sequential algorithms using active learning or Bayesian Optimization (Gutmann and Corander, 2016; Järvenpää et al., 2019; Lueckmann et al., 2019; Aushev et al., 2020), or 'gray-box' algorithms, which use additional information about or from the simulator (e.g., Baydin et al., 2019; Brehmer et al., 2020). We focused on approaches using neural networks for density estimation and did not compare to alternatives using Gaussian Processes (e.g., Meeds and Welling, 2014; Wilkinson, 2014). There are many other algorithms which the benchmark is currently lacking (e.g., Nott et al., 2014; Ong et al., 2018; Clarté et al., 2020; Prangle, 2019; Priddle et al., 2019; Picchini et al., 2020; Radev et al., 2020; Rodrigues et al., 2020). Keeping our initial selection small allowed us to carefully investigate hyperparameter choices. We focused on sequential algorithms with less sophisticated acquisition schemes and the black-box scenario, since we think these are important baselines for future comparisons.

Second, the tasks we considered do not cover the variety of possible challenges. Notably, while we have tasks with high dimensional data with and without structure, we have not included tasks with high-dimensional spatial structure, e.g., images. Such tasks would require algorithms that automatically learn summary statistics while exploring the structure of the data (e.g., Dinev and Gutmann, 2018; Greenberg et al., 2019; Hermans et al., 2020; Chen et al., 2021), an active research area.

Third, while we extensively investigated tuning choices and compared implementations, the results might nevertheless reflect our own areas of expertise.

Fourth, in line with common practice in SBI, results presented in the paper focused on performance as a function of the number of simulation calls. It is important to remember that differences in computation time can be substantial (see Appendix R): For example, (S)ABC was much faster than approaches requiring network training. Overall, sequential neural algorithms exhibited longest runtimes.

Fifth, for reasons described above, we focused on problems for which reference posteriors can be computed. This raises the question of how insights on these problems will generalize to 'real-world' simulators. Notably, even these simple problems already identify clear differences between, and limitations of, different SBI approaches. Since it is not possible to rigorously compare the performance of different algorithms directly on 'real-world' simulators due to the lack of appropriate metrics, we see the benchmark as a necessary stepping stone towards the development of (potentially automated) selection strategies for practical problems.

Sixth, in practice, the choice of algorithm can depend on aspects that are difficult to quantify: It will depend on the available information about a problem, the inference goal, and the speed of the simulator, among other considerations. We included some practical considerations and recommendations in Box 1.

Finally, benchmarking is an important tool, but not an end in itself—for example, conceptually new ideas might initially not yield competitive results but only reveal their true value later. Conversely, 'overfitting' on benchmarks can lead to the illusion of progress, and result in an undue focus on small implementation details which might not generalize beyond it. It would certainly be possible to cheat on this benchmark: In particular, as the simulators are available, one could use samples (or even likelihoods) to excessively tune hyperparameters *for each task*. This would hardly transfer to practice where such tuning is usually impossible (lack of metrics and expensive simulators). Therefore, we carefully compared choices and selected hyperparameters performing best *across tasks* (Appendix H).

5 Discussion

Quantitatively evaluating, comparing and improving algorithms through benchmarking is at the core of progress in machine learning. We here provided an initial benchmark for simulation-based inference. If used sensibly, it will be an important tool for clarifying and expediting progress in SBI. We hope that the current results on multiple widely-used algorithms already provide insights into the state of the field, assist researchers with algorithm development, and that our recommendations for practitioners will help them in selecting appropriate algorithms.

We believe that the full potential of the benchmark will be revealed as more researchers participate and contribute. To facilitate this process, and allow users to quickly explore and compare algorithms, we are providing precomputed reference posteriors, a website (`sbi-benchmark.github.io`), and open-source code (`github.com/sbi-benchmark/sbibm`).

Acknowledgements

We thank Álvaro Tejero-Cantero, Auguste Schulz, Conor Durkan, François Lanusse, Leandra White, Marcel Nonnenmacher, Michael Deistler, Pedro Rodrigues, Poornima Ramesh, Sören Becker and Theofanis Karaletsos for discussions and comments on the manuscript. In addition, J.-M.L. would like to thank the organisers and participants of the Likelihood-Free Inference Workshop hosted by the Simons Foundation for discussions, in particular, Danley Hsu, François Lanusse, George Papamakarios, Henri Pesonen, Joeri Hermans, Johann Brehmer, Kyle Cranmer, Owen Thomas and Umberto Simola. We also acknowledge and thank the Python (Van Rossum and Drake Jr, 1995) and Julia (Bezanson et al., 2017) communities for developing the tools enabling this work, including `Altair` (VanderPlas et al., 2018), `DifferentialEquations.jl` (Rackauckas and Nie, 2017), `Hydra` (Yadan, 2019), `kernel-gof` (Jitkrittum et al., 2017), `igms` (Sutherland, 2017), `NumPy` (Harris et al., 2020), `pandas` (pandas development team, 2020), `pyABC` (Klinger et al., 2018), `pyabcranger` (Collin et al., 2020), `Pyro` (Bingham et al., 2019), `PyTorch` (Paszke et al., 2019), `sbi` (Tejero-Cantero et al., 2020), `Scikit-learn` (Pedregosa et al., 2011), `torch-two-sample` (Djolonga, 2017), and `vega-lite` (Satyanarayan et al., 2017).

This work was supported by the German Research Foundation (DFG; SFB 1233 PN 276693517, SFB 1089, SPP 2041, Germany's Excellence Strategy – EXC number 2064/1 PN 390727645) and the German Federal Ministry of Education and Research (BMBF; project 'ADIMEM', FKZ 01IS18052 A-D).

References

Alsing, J., B. Wandelt, and S. Feeney
2018. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Monthly Notices of the Royal Astronomical Society*, 477(3):2874–2885.

Aushev, A., H. Pesonen, M. Heinonen, J. Corander, and S. Kaski
2020. Likelihood-free inference with deep gaussian processes. *Deep Learning and Inverse Problems Workshop at Neural Information Processing Systems*.

Baydin, A. G., L. Shao, W. Bhimji, L. Heinrich, L. Meadows, J. Liu, A. Munk, S. Naderiparizi, B. Gram-Hansen, G. Louppe, et al.
2019. Etalumis: bringing probabilistic programming to scientific simulators at scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Pp. 1–24.

Beaumont, M. A., J.-M. Cornuet, J.-M. Marin, and C. P. Robert
2009. Adaptive approximate bayesian computation. *Biometrika*, 96(4):983–990.

Beaumont, M. A., W. Zhang, and D. J. Balding
2002. Approximate bayesian computation in population genetics. *Genetics*, 162(4):2025–2035.

Bellemare, M. G., Y. Naddaf, J. Veness, and M. Bowling
2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Bezanson, J., A. Edelman, S. Karpinski, and V. B. Shah
2017. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

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

Blum, M. G. and O. François
2010. Non-linear regression models for approximate bayesian computation. *Statistics and Computing*, 20(1):63–73.

Brehmer, J., K. Cranmer, G. Louppe, and J. Pavez
2018. Constraining effective field theories with machine learning. *Physical Review Letters*, 121(11):111801.

Brehmer, J., G. Louppe, J. Pavez, and K. Cranmer
2020. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10):5242–5249.

Chan, J., V. Perrone, J. Spence, P. Jenkins, S. Mathieson, and Y. Song
2018. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Advances in Neural Information Processing Systems 31*, Pp. 8594–8605. Curran Associates, Inc.

Charnock, T., G. Lavaux, and B. D. Wandelt
2018. Automatic physical inference with information maximizing neural networks. *Physical Review D*, 97(8):083004.

Chen, Y., D. Zhang, M. Gutmann, A. Courville, and Z. Zhu
2021. Neural approximate sufficient statistics for implicit models. In *Proceedings of the 9th International Conference on Learning Representations, ICLR*.

Chwialkowski, K., H. Strathmann, and A. Gretton
2016. A kernel test of goodness of fit. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, Pp. 2606–2615. PMLR.

Clarté, G., C. P. Robert, R. J. Ryder, and J. Stoehr
2020. Component-wise approximate bayesian computation via gibbs-like steps. *Biometrika*.

Collin, F.-D., A. Estoup, J.-M. Marin, and L. Raynal
2020. Bringing abc inference to the machine learning realm: Abcranger, an optimized random forests library for abc. In *JOBIM 2020*, volume 2020.

Cranmer, K., J. Brehmer, and G. Louppe
2020. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*.

Cranmer, K., J. Pavez, and G. Louppe
2015. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*.

Dalmasso, N., A. B. Lee, R. Izbicki, T. Pospisil, and C.-A. Lin
2020. Validation of approximate likelihood and emulator models for computationally intensive simulations. In *Proceedings of The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Dinev, T. and M. U. Gutmann
2018. Dynamic likelihood-free inference via ratio estimation (dire). *arXiv preprint arXiv:1810.09899*.

Djolonga, J.
2017. torch-two-sample: A pytorch library for differentiable two-sample tests. Github.

Drovandi, C. C., C. Grazian, K. Mengersen, and C. Robert
2018. Approximating the likelihood in approximate bayesian computation. In *Handbook of Approximate Bayesian Computation*, S. Sisson, Y. Fan, and M. Beaumont, eds., chapter 12. CRC Press, Taylor & Francis Group.

Duan, Y., X. Chen, R. Houthooft, J. Schulman, and P. Abbeel
2016. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33th International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, Pp. 1329–1338. PMLR.

Durkan, C., A. Bekasov, I. Murray, and G. Papamakarios
2019. Neural spline flows. In *Advances in Neural Information Processing Systems*, Pp. 7509–7520. Curran Associates, Inc.

Durkan, C., I. Murray, and G. Papamakarios
2020. On contrastive learning for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 98 of *Proceedings of Machine Learning Research*. PMLR.

Durkan, C., G. Papamakarios, and I. Murray
2018. Sequential neural methods for likelihood-free inference. *Bayesian Deep Learning Workshop at Neural Information Processing Systems*.

Dutta, R., J. Corander, S. Kaski, and M. U. Gutmann
2016. Likelihood-free inference by ratio estimation. *arXiv preprint arXiv:1611.10242*.

Filos, A., S. Farquhar, A. N. Gomez, T. G. Rudner, Z. Kenton, L. Smith, M. Alizadeh, A. de Kroon, and Y. Gal
2019. A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks. *Bayesian Deep Learning Workshop at Neural Information Processing Systems*.

Friedman, J.
2004. On multivariate goodness-of-fit and two-sample testing. In *Conference on Statistical Problems in Particle Physics, Astrophysics and Cosmology*.

Gonçalves, P. J., J.-M. Lueckmann, M. Deistler, M. Nonnenmacher, K. Öcal, G. Bassetto, C. Chintaluri, W. F. Podlaski, S. A. Haddad, T. P. Vogels, D. S. Greenberg, and J. H. Macke
2020. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*.

Gourieroux, C., A. Monfort, and E. Renault
1993. Indirect inference. *Journal of Applied Econometrics*, 8(S1):S85–S118.

Greenberg, D., M. Nonnenmacher, and J. Macke
2019. Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, Pp. 2404–2414. PMLR.

Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola
2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(Mar):723–773.

Gutmann, M. U. and J. Corander
2016. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *The Journal of Machine Learning Research*, 17(1):4256–4302.

Gutmann, M. U., R. Dutta, S. Kaski, and J. Corander
2018. Likelihood-free inference via classification. *Statistics and Computing*, 28(2):411–425.

Harris, C. R., K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al.
2020. Array programming with numpy. *Nature*, 585(7825):357–362.

Hermans, J., V. Begy, and G. Louppe
2020. Likelihood-free mcmc with approximate likelihood ratios. In *Proceedings of the 37th International*

Conference on Machine Learning, volume 98 of *Proceedings of Machine Learning Research*. PMLR.

Hirsch, H.-G. and D. Pearce
2000. The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ASR2000-Automatic Speech Recognition: Challenges for the new Millenium ISCA Tutorial and Research Workshop (ITRW)*.

Hogg, D. W. and D. Foreman-Mackey
2018. Data analysis recipes: Using markov chain monte carlo. *The Astrophysical Journal Supplement Series*, 236(1):11.

Izbicki, R., A. Lee, and C. Schafer
2014. High-dimensional density ratio estimation with extensions to approximate likelihood computation. In *Artificial Intelligence and Statistics*, Pp. 420–429.

Järvenpää, M., M. U. Gutmann, A. Pleska, A. Vehtari, P. Marttinen, et al.
2019. Efficient acquisition rules for model-based approximate bayesian computation. *Bayesian Analysis*, 14(2):595–622.

Järvenpää, M., M. U. Gutmann, A. Vehtari, P. Marttinen, et al.
2020. Parallel gaussian process surrogate bayesian inference with noisy likelihood evaluations. *Bayesian Analysis*.

Jitkrittum, W., W. Xu, Z. Szabó, K. Fukumizu, and A. Gretton
2017. A linear-time kernel goodness-of-fit test. In *Advances in Neural Information Processing Systems*, Pp. 262–271.

Karabatsos, G. and F. Leisen
2018. An approximate likelihood perspective on abc methods. *Statistics Surveys*, 12:66–104.

Klinger, E., D. Rickert, and J. Hasenauer
2018. pyabc: distributed, likelihood-free inference. *Bioinformatics*, 34(20):3591–3593.

Le, T. A., A. G. Baydin, and F. Wood
2017. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54. JMLR.

Liu, F., W. Xu, J. Lu, G. Zhang, A. Gretton, and D. J. Sutherland
2020. Learning deep kernels for non-parametric two-sample tests. In *Proceedings of the 37th International Conference on Machine Learning*, volume 98 of *Proceedings of Machine Learning Research*. PMLR.

Liu, Q., J. Lee, and M. Jordan
2016. A kernelized stein discrepancy for goodness-of-fit tests. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, Pp. 276–284. PMLR.

Lopez-Paz, D. and M. Oquab
2017. Revisiting classifier two-sample tests. In *5th International Conference on Learning Representations, ICLR*.

Lueckmann, J.-M., G. Bassetto, T. Karaletsos, and J. H. Macke
2019. Likelihood-free inference with emulator networks. In *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96 of *Proceedings of Machine Learning Research*, Pp. 32–53. PMLR.

Lueckmann, J.-M., P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke
2017. Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in Neural Information Processing Systems 30*, Pp. 1289–1299. Curran Associates, Inc.

Marjoram, P. and S. Tavaré
2006. Modern computational approaches for analysing molecular genetic variation data. *Nature Reviews Genetics*, 7(10):759–770.

Meeds, E. and M. Welling
2014. Gps-abc: Gaussian process surrogate approximate bayesian computation. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, P. 593–602, Arlington, Virginia, USA. AUAI Press.

Nott, D. J., Y. Fan, L. Marshall, and S. Sisson
2014. Approximate bayesian computation and bayes' linear analysis: toward high-dimensional abc. *Journal of Computational and Graphical Statistics*, 23(1):65–86.

Ong, V. M.-H., D. J. Nott, M.-N. Tran, S. A. Sisson, and C. C. Drovandi
2018. Likelihood-free inference in high dimensions with synthetic likelihood. *Computational Statistics & Data Analysis*, 128:271 – 291.

pandas development team, T.
2020. pandas-dev/pandas: Pandas.

Papamakarios, G. and I. Murray
2016. Fast ϵ-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems 29*, Pp. 1028–1036. Curran Associates, Inc.

Papamakarios, G., E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan
2019a. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

Papamakarios, G., T. Pavlakou, and I. Murray
2017. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems 30*, Pp. 2338–2347. Curran Associates, Inc.

Papamakarios, G., D. Sterratt, and I. Murray
2019b. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, Pp. 837–848. PMLR.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala
2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, Pp. 8024–8035. Curran Associates, Inc.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay
2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pham, K. C., D. J. Nott, and S. Chaudhuri
2014. A note on approximating abc-mcmc using flexible classifiers. *Stat*, 3(1):218–227.

Picchini, U., U. Simola, and J. Corander
2020. Adaptive mcmc for synthetic likelihoods and correlated synthetic likelihoods. *arXiv preprint arXiv:2004.04558*.

Prangle, D.
2019. Distilling importance sampling. *arXiv preprint arXiv:1910.03632*.

Prangle, D., P. Fearnhead, M. P. Cox, P. J. Biggs, and N. P. French
2014. Semi-automatic selection of summary statistics for abc model choice. *Statistical applications in genetics and molecular biology*, 13(1):67–82.

Priddle, J. W., S. A. Sisson, and C. Drovandi
2019. Efficient bayesian synthetic likelihood with whitening transformations. *arXiv preprint arXiv:1909.04857*.

Pritchard, J. K., M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman
1999. Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798.

Rackauckas, C. and Q. Nie
2017. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *The Journal of Open Research Software*, 5(1).

Radev, S. T., U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe
2020. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Ramdas, A., S. J. Reddi, B. Poczos, A. Singh, and L. Wasserman
2015. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. *AAAI Conference on Artificial Intelligence*.

Ratmann, O., O. Jørgensen, T. Hinkley, M. Stumpf, S. Richardson, and C. Wiuf
2007. Using likelihood-free inference to compare evolutionary dynamics of the protein networks of h. pylori and p. falciparum. *PLoS Computational Biology*, 3(11).

Raynal, L., J.-M. Marin, P. Pudlo, M. Ribatet, C. P. Robert, and A. Estoup
2019. Abc random forests for bayesian parameter inference. *Bioinformatics*, 35(10):1720–1728.

Rezende, D. and S. Mohamed
2015. Variational inference with normalizing flows. In *Proceedings of The 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, Pp. 1530–1538. PMLR.

Rios, L. M. and N. V. Sahinidis
2013. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293.

Rodrigues, G., D. J. Nott, and S. Sisson
2020. Likelihood-free approximate gibbs sampling. *Statistics and Computing*, Pp. 1–17.

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al.
2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Satyanarayan, A., D. Moritz, K. Wongsuphasawat, and J. Heer
2017. Vega-lite: A grammar of interactive graphics. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*.

Shahriari, B., K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas
2015. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Simola, U., J. Cisewski-Kehe, M. U. Gutmann, J. Corander, et al.
2020. Adaptive approximate bayesian computation tolerance selection. *Bayesian Analysis*.

Sisson, S. A., Y. Fan, and M. M. Tanaka
2007. Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765.

Sisson, S. A., F. Y., and B. M. A.
2018. Overview of abc. In *Handbook of Approximate Bayesian Computation*, chapter 1. CRC Press, Taylor & Francis Group.

Sutherland, D. J.
2017. igms: Implicit generative models. Github.

Sutherland, D. J., H.-Y. Tung, H. Strathmann, S. De, A. Ramdas, A. Smola, and A. Gretton
2017. Generative models and model criticism via optimized maximum mean discrepancy. *5th International Conference on Learning Representations, ICLR*.

Talts, S., M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman
2018. Validating bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*.

Tavaré, S., D. J. Balding, R. C. Griffiths, and P. Donnelly
1997. Inferring coalescence times from dna sequence data. *Genetics*, 145(2).

Tejero-Cantero, A., J. Boelts, M. Deistler, J.-M. Lueckmann, C. Durkan, P. J. Gonçalves, D. S. Greenberg, and J. H. Macke
2020. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505.

Thomas, O., R. Dutta, J. Corander, S. Kaski, and M. U. Gutmann
2020. Likelihood-free inference by ratio estimation. *Bayesian Analysis*.

Toni, T., D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf
2009. Approximate bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of the Royal Society Interface*, 6(31):187–202.

Van Rossum, G. and F. L. Drake Jr
1995. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands.

VanderPlas, J., B. E. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert
2018. Altair: Interactive statistical visualizations for python. *The Journal of Open Source Software*, 3(32).

Wang, A., A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman
2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Pp. 353–355. Association for Computational Linguistics.

Wenzel, F., K. Roth, B. S. Veeling, J. Świątkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin
2020. How good is the bayes posterior in deep neural networks really? In *Proceedings of the 37th International Conference on Machine Learning*, volume 98 of *Proceedings of Machine Learning Research*. PMLR.

Wilkinson, R. D.
2014. Accelerating abc methods using gaussian processes. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33 of *Proceedings of Machine Learning Research*, Pp. 1015–1023. PMLR.

Wood, F., A. Warrington, S. Naderiparizi, C. Weilbach, V. Masrani, W. Harvey, A. Scibior, B. Beronov, and A. Nasseri
2020. Planning as inference in epidemiological models. *arXiv preprint arXiv:2003.13221*.

Wood, S. N.
2010. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104.

Yadan, O.
2019. Hydra - a framework for elegantly configuring complex applications. Github.