Adaptive Gaussian Copula ABC

Yanzhi Chen
The University of Edinburgh

Michael U. Gutmann
The University of Edinburgh

Abstract

Approximate Bayesian computation (ABC) is a set of techniques for Bayesian inference when the likelihood is intractable but sampling from the model is possible. This work presents a simple yet effective ABC algorithm based on the combination of two classical ABC approaches — regression ABC and sequential ABC. The key idea is that rather than learning the posterior directly, we first target another auxiliary distribution that can be learned accurately by existing methods, through which we then subsequently learn the desired posterior with the help of a Gaussian copula. During this process, the complexity of the model changes adaptively according to the data at hand. Experiments on a synthetic dataset as well as three real-world inference tasks demonstrates that the proposed method is fast, accurate, and easy to use.

1 Introduction

Many parametric statistical models are specified in terms of a parametrised data generating process. We can sample or simulate data from this kind of models but their likelihood function is typically too costly to evaluate. The models are called implicit (Diggle and Gratton, 1984) or simulator-based models (Gutmann and Corander, 2016) and are widely used in scientific domains including ecology (Wood, 2010), epidemiology (Corander et al., 2017), psychology (Karabatsos, 2017) and cosmology (Weyant et al., 2013). For example, the demographic evolution of two species can be simulated by a set of stochastic differential equations controlled by birth/predation rates but computation of the likelihood is intractable. Often, the true interest is rarely in simulating data, but in the inverse problem of identifying model parameters that could have plausibly produced the observed data. This usually not only includes point estimates of the model parameters but also a quantification of their uncertainty.

The above task can, in principle, be solved by the well-known framework of Bayesian inference. However, due to the absence of likelihood functions for implicit models, exact inference becomes difficult or even impossible. The technique of approximate Bayesian computation (ABC) enables inference in such circumstances (for recent reviews, see e.g. Lintusaari et al., 2017; Sisson et al., 2018). The most basic ABC algorithm works by repeatedly simulating data with different parameters and only accepting those parameters whose simulated data 'resemble' the observed ones. This generally requires hitting a small ϵ-ball for good accuracy, which is unfortunately impractical in high-dimensional spaces due to the well-known curse of dimensionality (see e.g. Beaumont, 2010).

Several efforts have been made to alleviate the above accuracy-efficiency dilemma in ABC (e.g. Marjoram et al., 2003; Beaumont et al., 2002, 2009; Blum and François, 2010; Gutmann and Corander, 2016; Papamakarios and Murray, 2016). The first line of work, regression ABC (Beaumont et al., 2002; Blum and François, 2010), accelerates inference by first allowing a much larger ϵ-ball, and then capturing the relationship between the simulated data and the parameters inside this ϵ-ball through a regression function, with which accepted samples are post-adjusted so as to compensate for the accuracy loss due to the larger ϵ used. Another line of work improves the efficiency of ABC by learning the posterior iteratively (Sisson et al., 2007; Beaumont et al., 2009; Papamakarios and Murray, 2016; Lueckmann et al., 2017). In these methods, preliminary models of the posterior are used to guide further simulations, leading to a significant drop of overall simulation costs. Both lines of methods are shown to be effective in a wide range of inference tasks.

In this paper, we show how these two lines of works

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

can refine each other, yielding a powerful yet simple algorithm. We bridge the two lines of work using Gaussian copula modelling (Li et al., 2017; Fan et al., 2013). The resulting proposed approach, which we call adaptive Gaussian copula ABC (AGC-ABC), has similar power as recent flexible machine-learning methods (Papamakarios and Murray, 2016; Lueckmann et al., 2017) but requires less simulated data. This naturally leads to performance gains for smaller computational budgets, as we confirm empirically.

The rest of the paper is organized as follows. Section 2 gives an overview of regression ABC and sequential ABC. Section 3 details the proposed AGC-ABC algorithm. Section 4 compares the proposed method to existing works and Section 5 contains the experimental results. Section 6 concludes the paper.

2 Background

Given observed data $\mathbf{x}^o$, the task of approximate Bayesian inference for implicit models is to estimate the posterior density of the parameters $\boldsymbol{\theta}$ without evaluating the likelihood directly. To achieve this goal, methods known as approximate Bayesian computation (ABC) typically reduce the observed data to summary statistics $\mathbf{s}^o = \mathbf{s}(\mathbf{x}^o)$ and approximate the likelihood by $\mathbb{P}(\|\mathbf{s} - \mathbf{s}^o\| < \epsilon |\ \boldsymbol{\theta})$ where ϵ is chosen sufficiently small. The approximate posterior is then

$$\pi(\boldsymbol{\theta}|\mathbf{s}^o) \propto \pi(\boldsymbol{\theta}) \cdot \mathbb{P}(\|\mathbf{s} - \mathbf{s}^o\| < \epsilon|\boldsymbol{\theta}). \tag{1}$$

ABC algorithms typically only yield samples from the above approximate posterior. The most basic algorithm, ABC rejection sampling, first proposes $\boldsymbol{\theta}$ from the prior $\pi(\boldsymbol{\theta})$ and then only accepts those $\boldsymbol{\theta}$ whose simulated summary statistics fall within an ϵ-ball $\mathrm{B}_\epsilon(\mathbf{s}^o) = \{\mathbf{s} : \|\mathbf{s} - \mathbf{s}^o\| \le \epsilon\}$ around $\mathbf{s}^o$. While simple and robust, the algorithm suffers from a poor trade-off between accuracy and computational efficiency (see e.g. Lintusaari et al., 2017). Two lines of work aim at improving the trade-off from different perspectives: regression ABC and sequential ABC.

2.1 Regression ABC

This line of work first allows a much larger ϵ-ball in rejection ABC, and then uses the accepted samples to estimate the relationship between $\boldsymbol{\theta}$ and $\mathbf{s}$ through regression. The learned regression model is then used to adjust the samples so that, if the regression model is correct, the adjusted samples correspond to samples from $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$ in (1) with $\epsilon = 0$.

Conditional on the summary statistics $\mathbf{s}$, an additive noise model is typically used to describe the relationship between $\boldsymbol{\theta}$ and $\mathbf{s}$ as

$$\boldsymbol{\theta}|\mathbf{s} = \underbrace{\mathbb{E}[\boldsymbol{\theta}|\mathbf{s}]}_{\text{mean}} + \underbrace{\boldsymbol{\xi}|\mathbf{s}}_{\text{residual}}. \tag{2}$$

Here, we used the notation $\boldsymbol{\theta}|\mathbf{s}$ and $\boldsymbol{\xi}|\mathbf{s}$ to highlight that the above relationship is conditional on $\mathbf{s}$. In particular, the distribution of the residual may depend on the value of $\mathbf{s}$ so that $p(\boldsymbol{\xi}|\mathbf{s}) \neq p(\boldsymbol{\xi})$. Under the above model, to sample from the posterior $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$, we need to evaluate $\mathbb{E}[\boldsymbol{\theta}|\mathbf{s}^o]$ and sample from the conditional distribution of $\boldsymbol{\xi}$ at $\mathbf{s}^o$.

To learn $\mathbb{E}[\boldsymbol{\theta}|\mathbf{s}]$, regression ABC fits a regression function $g(\cdot)$ on the samples $(\boldsymbol{\theta}^{(i)}, \mathbf{s}^{(i)})$, $i = 1, \ldots, I$, collected in the preliminary run of rejection ABC. The fitting is typically done by the method of least squares, i.e. by minimising

$$J(g) = \frac{1}{I} \sum_{i=1}^{I} \|\boldsymbol{\theta}^{(i)} - g(\mathbf{s}^{(i)})\|_2^2 \tag{3}$$

since it is simple and, for sufficiently many samples (training data), the optimal $g(\mathbf{s})$ is guaranteed to approach $\mathbb{E}[\boldsymbol{\theta}|\mathbf{s}]$. A critical aspect of this approach is the choice of the function family among which one searches for the optimal g. For small amounts of training data, linear regression (Beaumont et al., 2002) is a safe choice while more flexible neural networks (Blum and François, 2010) may be used when more training data are available.

On obtaining $g(\cdot)$ we get an approximation for the posterior mean at $\mathbf{s}^o$ via $g(\mathbf{s}^o)$. What remains unknown is the distribution $p(\boldsymbol{\xi}|\mathbf{s}^o)$. To estimate this, regression ABC makes the *homogeneity* assumption that the distribution of the residual remains roughly unchanged for all $\mathbf{s}$ in the ϵ-ball $\mathrm{B}_\epsilon(\mathbf{s}^o)$ around $\mathbf{s}^o$:

$$p(\boldsymbol{\xi}|\mathbf{s}) \approx p(\boldsymbol{\xi}) \qquad \forall \mathbf{s} \in \mathrm{B}_\epsilon(\mathbf{s}^o) \tag{4}$$

With this assumption, the empirical residuals $\boldsymbol{\theta}^{(i)} - g(\mathbf{s}^{(i)})$ are used as samples from $p(\boldsymbol{\xi}|\mathbf{s}^o)$, and the adjustment

$$\boldsymbol{\theta}'^{(i)} = \underbrace{g(\mathbf{s}^o)}_{\text{est.mean}} + \underbrace{\boldsymbol{\theta}^{(i)} - g(\mathbf{s}^{(i)})}_{\text{est.residual}} \tag{5}$$

yields samples from $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$ in (1) with $\epsilon = 0$. However, this result hinges on (a) the mean function being learned well and (b) the homogeneity assumption holding so that the estimated residuals do correspond to samples from $p(\boldsymbol{\xi}|\mathbf{s}^o)$. For a theoretical analysis of regression ABC, see e.g (Blum, 2010).

2.2 Sequential ABC

This second line of work aims at accelerating the inference by learning the posterior in an iterative way

(Beaumont et al., 2009; Sisson et al., 2007; Papamakarios and Murray, 2016; Lueckmann et al., 2017). Using the prior $\pi(\boldsymbol{\theta})$ as proposal distribution in the generation of the parameter-data tuples $(\boldsymbol{\theta}^{(i)}, \mathbf{s}^{(i)})$, the methods first learn a rough but computationally cheaper approximation $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ to the posterior, e.g. by using a larger ϵ value (Beaumont et al., 2009; Sisson et al., 2007), or a small fraction of the overall simulation budget (Papamakarios and Murray, 2016; Lueckmann et al., 2017). The learned posterior $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ is then used as a new proposal distribution (new 'prior') $p(\boldsymbol{\theta})$ in the next round. From newly generated tuples $(\boldsymbol{\theta}^{(i)}, \mathbf{s}^{(i)})$, with $\boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta})$, the methods then construct an auxiliary posterior $p(\boldsymbol{\theta}|\mathbf{s}^o)$. In order to correct for using another proposal distribution than the prior, the auxiliary posterior $p(\boldsymbol{\theta}|\mathbf{s}^o)$ and samples from it need to be reweighted by

$$w(\boldsymbol{\theta}) \propto \frac{\pi(\boldsymbol{\theta})}{p(\boldsymbol{\theta})} \tag{6}$$

see e.g. (Lintusaari et al., 2017). A reason why this method accelerates inference is that the new, more informed, proposal $p(\boldsymbol{\theta})$ generates more often summary statistics $\mathbf{s}$ that fall inside the ϵ-ball $\mathrm{B}_\epsilon(\mathbf{s}^o)$. For a given ϵ, this then results in a higher acceptance rate and a reduction of the simulation cost.

The concrete algorithms of sequential ABC may differ across methods. Earlier sequential Monte Carlo ABC methods (SMC-ABC: Beaumont et al., 2009; Sisson et al., 2007) adopt a non-parametric posterior modeling, whereas the recent sequential neural posterior ABC ones (SNP-ABC: Papamakarios and Murray, 2016; Lueckmann et al., 2017) take a parametric approach using mixture density networks. However, the underlying basic principles remain roughly the same.

3 Methodology

We here explain the proposed approach that makes use of the basic principles of both regression ABC and sequential ABC.

The performance of regression ABC is likely to suffer if the homogeneity assumption in (4) does not hold. If ABC is run with a broad proposal distribution (e.g. the prior), only very few of the simulated $\mathbf{s}$ are close to $\mathbf{s}^o$, so that deviation from the homogeneity is generally likely (Figure 1.a). However, if we use a more informed proposal distribution, more samples of $\mathbf{s}$ around $\mathbf{s}^o$ can be collected (Figure 1.b). In other words, the shape of the proposal distribution strongly affects whether the homogeneity assumption in regression ABC reasonably holds. This motivates us to take a two-stage approach where, in a first coarse-grained phase, we learn a rough approximation to the posterior and then, in a subsequent fine-grained-phase, use it as proposal distribution in order to generate a sufficient amount of $(\boldsymbol{\theta}, \mathbf{s})$ pairs for which the homogeneity assumption in regression ABC is better satisfied.

Regression function. We will be using regression-ABC in both the coarse- and fine-grained phase, for which we need to choose a family of regression functions g. Simple regression models with linear functions g are easy to learn but are not able to capture possible nonlinear relationships between $\mathbf{s}$ and $\boldsymbol{\theta}$. On the other hand, neural networks are more flexible; however they also require more training data. Here, we thus perform model selection by comparing the validation error between linear regression and a neural network trained with early stopping. The split into training/validation set is done on a 80/20 basis. Before regression, each $\boldsymbol{\theta}$ is preprocessed such that they have the same scale along each dimension.

Coarse-grained phase. In this stage we learn a rough but robust approximation $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ to the posterior using only a small fraction λ of the overall simulation budget N. We do this by performing ordinary regression ABC based on parameters generated from the prior $\pi(\boldsymbol{\theta})$. We then pick a subset of the adjusted samples $\{\boldsymbol{\theta}'^{(1)}, ..., \boldsymbol{\theta}'^{(m)}\}$ whose corresponding summary statistics $\mathbf{s}$ are among the top m closest to $\mathbf{s}^o$ and model the adjusted samples with a Gaussian distribution so that

$$\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o) = \mathcal{N}\Big(\boldsymbol{\theta}; g(\mathbf{s}^o), \mathbf{V}\Big). \tag{7}$$

Here, $g(\cdot)$ is the learned regression function and $\mathbf{V}$ is the inflated sample covariance matrix computed as $\mathbf{V} = \alpha \cdot \frac{1}{m}\sum_i^m [\boldsymbol{\theta}'^{(i)} - g(\mathbf{s}^o)][\boldsymbol{\theta}'^{(i)} - g(\mathbf{s}^o)]^\top$, $\alpha = 1.5$. By using a Gaussian, we work with the maximum entropy distribution given mean and covariance of the adjusted samples. The motivation for inflating the covariance matrix is that a slightly broader proposal distribution is more robust to e.g. estimation errors in the learned regression function (Figure 1.c).

For the value of m, we choose m so that the top 20% summary statistics closest to $\mathbf{s}^o$ are retained, which is a conservative but robust strategy given the small simulation budget in this phase. The whole coarse-grained learning procedure is summarized in Algorithm 1, where, for simplicity, we incorporated the reduction to summary statistics directly into the data generating process (simulator).

Fine-grained phase. After learning $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ we use it as proposal distribution $p(\boldsymbol{\theta})$ in another round of regression ABC with the remaining $1 - \lambda$ simulation budget. We adjust the top n samples whose summary statistics $\mathbf{s}$ are closest to $\mathbf{s}^o$ and then model their distribution as a Gaussian copula.

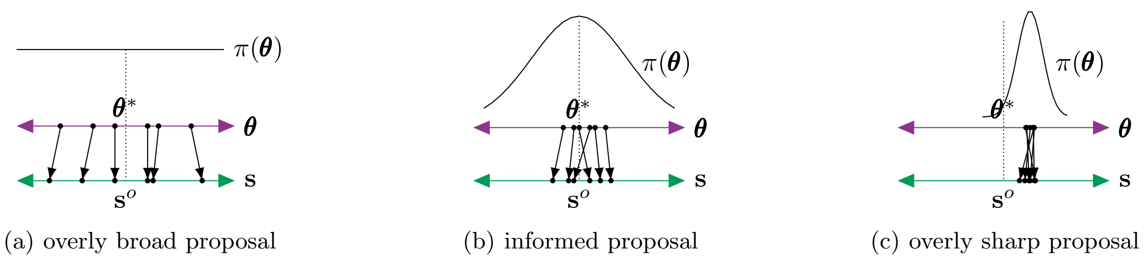

Figure 1: The effect of different proposal distributions in regression ABC.

Gaussian copula are a generalisation of the Gaussian distribution. They can be seen to model (the adjusted) $\boldsymbol{\theta}' = (\theta'_1, \ldots, \theta'_K)^\top$ in terms of latent variables $\mathbf{z}$ that are element-wise non-linearly transformed to give $\theta'_k = F_k^{-1}(\Phi(z_k))$ where F_k is the marginal cumulative distribution function (CDF) of θ'_k and $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{\Lambda})$. Under this construction, the auxiliary posterior $p(\boldsymbol{\theta}|\mathbf{s}^o)$ of the adjusted samples equals

$$p(\boldsymbol{\theta}|\mathbf{s}^o) = \underbrace{\frac{\mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{\Lambda})}{\prod \mathcal{N}(z_k; 0, 1)}}_{c_{\mathrm{GC}}(u_1, \ldots, u_k; \Lambda)} \cdot \left[\prod_k^K f_k(\theta_k) \right], \tag{8}$$

where $\mathbf{\Lambda}$ is the correlation matrix whose diagonal elements are all ones and $c_{\mathrm{GC}}(\cdot)$ is called the Gaussian copula density.

The learning of Gaussian copula can be done in a semi-parametric way. Denote the adjusted samples by $\{\boldsymbol{\theta}'^{(1)}, ..., \boldsymbol{\theta}'^{(n)}\}$:

- *Marginal distribution.* Each $f_k(\theta_k)$ is learned by kernel density estimation (KDE), which is well-known to be typically accurate in 1D (Scott, 2015).
- *Correlation matrix.* The matrix $\mathbf{\Lambda}$ is learned by converting each sample $\boldsymbol{\theta}^{'(j)}$ to its corresponding latent representation $\mathbf{z}^{(j)}$ as $z_k^{(j)} = \Phi^{-1}(r_k(\theta_k^{'(j)}))$ where $r_k(x) = \frac{1}{n} \sum_{j=1}^n \mathbf{1}[\theta_k^{'(j)} < x]$, followed by estimating $\mathbf{\Lambda}$ as $\mathbf{\Lambda} = \frac{1}{n} \sum_j^n \mathbf{z}^{(j)} \mathbf{z}^{(j)\top}$.

As for sequential ABC in (6), we need to convert the auxiliary posterior $p(\boldsymbol{\theta}|\mathbf{s}^o)$ to an estimate of the actual posterior $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$ by reweighing it by the ratio of the prior to proposal distribution. This gives

$$\pi(\boldsymbol{\theta}|\mathbf{s}^o) \propto \frac{\pi(\boldsymbol{\theta})}{\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)} p(\boldsymbol{\theta}|\mathbf{s}^o), \tag{9}$$

where the proposal distribution $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ was defined in (7) and $p(\boldsymbol{\theta}|\mathbf{s}^o)$ was defined in (8). Note that while the auxiliary posterior $p(\boldsymbol{\theta}|\mathbf{s}^o)$ is a Gaussian copula, the reweighted $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$ is generally not (because of e.g. the presence of the prior).

Unlike in the coarse-grained phase, in this phase, we select n such that the top 2000 summary statistics closest to $\mathbf{s}^o$ are retained. We adopt this more aggressive strategy because here we have a larger simulation budget and better satisfy the homogeneity condition. We are therefore safe to keep a larger number of samples so as to reduce the Monte Carlo error. In addition, by retaining a fixed number of samples we effectively shrink the value of ϵ when the overall simulation budget N is increased. The whole fine-grained phase is summarized in Algorithm 2.

4 Comparisons to other methods

Here we compare AGC-ABC with three methods in the field: neural network regression ABC (NN-ABC: Blum and François, 2010), Gaussian copula ABC (GC-ABC: Li et al., 2017), and sequential neural posterior ABC (SNP-ABC: Papamakarios and Murray, 2016; Lueckmann et al., 2017).

AGC-ABC vs. NN-ABC. Both AGC-ABC and NN-ABC address the homogeneity issue (4) in regression ABC. NN-ABC aims to model possible heterogeneity of the residuals by taking into account the possibly changing scale of the residuals. In contrast to this, AGC-ABC uses a two-stage approach to stay in the homogeneous regime, where not only the scale but also the dependency structure of the residuals is modelled.

AGC-ABC vs. GC-ABC. Both AGC-ABC and GC-ABC make use of Gaussian copula in posterior modeling. GC-ABC can be viewed as a special case of AGC-ABC without the adapting coarse-grained stage. The posterior in GC-ABC is a Gaussian copula whereas ours is generally not.

AGC-ABC vs. SNP-ABC. Both AGC-ABC and SNP-ABC learn the posterior in a coarse-to-fine fashion. The difference lies in the model complexity. SNP-ABC models the relationship between $\boldsymbol{\theta}$ and $\mathbf{s}$ with a mixture of Gaussians whose parameters are computed by a neural network. While such constructions can approximate the posterior more and more accu-

Algorithm 1: `Coarse-grained learning`

```
Input: prior $\pi(\boldsymbol{\theta})$, observed data $\mathbf{s}^o$, simulation
  budget $\lambda N$, simulator $M(\boldsymbol{\theta})$
Output: coarse approximation $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$
Hyperparams: $m = \lceil 0.2\lambda N \rceil$.

for $i = 1$ to $\lceil \lambda N \rceil$ do
    sample $\boldsymbol{\theta}^{(i)} \sim \pi(\boldsymbol{\theta})$ ;
    simulate $\mathbf{s}^{(i)} \sim M(\boldsymbol{\theta})$ ;
end
fit $g = \arg\min_{\tilde{g}} J(\tilde{g})$ by (3) with $\{(\boldsymbol{\theta}^{(i)}, \mathbf{s}^{(i)})\}_{i=1}^{m}$ ;
$\boldsymbol{\theta}'^{(i)} \leftarrow g(\mathbf{s}^o) + \boldsymbol{\theta}^{(i)} - g(\mathbf{s}^{(i)})$, for $\forall i$;
sort $\boldsymbol{\theta}'^{(i)}$ by $\|\mathbf{s}^{(i)} - \mathbf{s}^o\|$ in ascendant order ;

$\boldsymbol{\mu} \leftarrow g(\mathbf{s}^o)$ ;
$\mathbf{V} \leftarrow \frac{3}{2m} \sum_{i=1}^{m} (\boldsymbol{\theta}'^{(i)} - \boldsymbol{\mu})(\boldsymbol{\theta}'^{(i)} - \boldsymbol{\mu})^\top$ ;
construct $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ with $(\boldsymbol{\mu}, \mathbf{V})$ as in (7) ;
return $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$.
```

Algorithm 2: `Fine-grained learning`

```
Input: prior $\pi(\boldsymbol{\theta})$, proposal $\tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$, observed data
  $\mathbf{s}^o$, simulation budget $(1-\lambda)N$, simulator $M(\boldsymbol{\theta})$
Output: estimated posterior $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$
Hyperparams: $n = 2000$.

for $i = 1$ to $\lfloor (1-\lambda)N \rfloor$ do
    sample $\boldsymbol{\theta}^{(i)} \sim \tilde{\pi}(\boldsymbol{\theta}|\mathbf{s}^o)$ ;
    simulate $\mathbf{s}^{(i)} \sim M(\boldsymbol{\theta})$ ;
end
fit $g = \arg\min_{\tilde{g}} J(\tilde{g})$ by (3) with $\{(\boldsymbol{\theta}^{(i)}, \mathbf{s}^{(i)})\}_{i=1}^{n}$ ;
$\boldsymbol{\theta}'^{(i)} \leftarrow g(\mathbf{s}^o) + \boldsymbol{\theta}^{(i)} - g(\mathbf{s}^{(i)})$, for $\forall i$;
sort $\boldsymbol{\theta}'^{(i)}$ by $\|\mathbf{s}^{(i)} - \mathbf{s}^o\|$ in ascendant order ;

$f_k(\theta) \leftarrow \texttt{KDE}(\{\theta_k'^{(1)}, ..., \theta_k'^{(n)}\})$, $\forall k$ ;
$\boldsymbol{\Lambda} \leftarrow \frac{1}{n} \sum_{i=1}^{n} \mathbf{z}^{(i)}\mathbf{z}^{(i)\top}$ with $z_k^{(i)} = \Phi^{-1}(r_k(\theta_k'^{(i)}))$ ;
construct $p(\boldsymbol{\theta}|\mathbf{s}^o)$ with $(\{f_k(\theta_k)\}_{k=1}^{K}, \boldsymbol{\Lambda})$ as in (8) ;
return $\pi(\boldsymbol{\theta}|\mathbf{s}^o)$ as in (9)
```

rately as the number of mixture components increases, the number of neural network parameters is generally much larger than in AGC-ABC where we only need to estimate the posterior mean and a correlation matrix. Additionally, the marginal distributions in our copula-based method can generally be modelled more accurately than in SNP-ABC, since we learn them non-parametrically for each dimension.

5 Experiments

5.1 Setup

We compare the proposed adaptive Gaussian Copula ABC (AGC-ABC) with five ABC methods in the field: rejection ABC (Pritchard et al., 1999), regression ABC (Beaumont et al., 2002), GC-ABC (Li et al., 2017), NN-ABC (Blum and François, 2010) and SNP-ABC (Papamakarios and Murray, 2016). Comparisons are done by computing the JSD between the true and the approximate posterior, and when the true posterior is not available, we approximate it by running a ABC rejection sampling algorithm with (a) well chosen summary statistics $\mathbf{s}$ and (b) an extremely small ϵ value e.g the 0.00001 quantile of the population of $\|\mathbf{s} - \mathbf{s}^o\|$, and then estimate it by KDE. Similarly, when the posterior approximated in each method is not available in analytical form (e.g for REJ-ABC, REG-ABC and NN-ABC), we estimate it by KDE. For two distributions P and Q, the JSD is given by

$$\mathrm{JSD}(P, Q) = \frac{1}{2}\mathrm{KL}\Big[P \| \frac{P+Q}{2}\Big] + \frac{1}{2}\mathrm{KL}\Big[Q \| \frac{P+Q}{2}\Big].$$

The calculation of JSD is done numerically by a Riemann sum over 30^K equally spaced grid points with K being the dimensionality of $\boldsymbol{\theta}$. The region of the grids is defined by the minimal and maximal values of the samples from P and Q jointly. Before the JSD calculation, all distributions are further re-normalized so that they sum to unity over the grid.

To make the comparison sensible, each method has the same total number of simulations (budget) N available. Note, however, that the different methods may use the simulation budget in different ways:

- For rejection ABC we pick the top 2000 samples closest to $\mathbf{s}^o$ for posterior estimation;
- For the two linear regression-based methods, regression ABC and GC-ABC, we pick the top 2000 samples for posterior learning;
- For NN-ABC and SNP-ABC we use all samples for learning, in line with their goal of modelling the relationship between $\mathbf{s}$ and $\boldsymbol{\theta}$ without restriction to an ϵ-ball.
- For AGC-ABC, as mentioned before, we use the top 20% of the samples in the coarse-grained phase while the top 2000 samples in the fine-grained phase.

Both AGC-ABC and the SNP-ABC learn the posteriors in a coarse-to-fine fashion. For these two methods, we assign 20%/80% of the overall simulation budget N to the coarse/fine phase respectively. Following the original literature (Papamakarios and Murray, 2016), the neural networks in SNP-ABC have one/two hidden layers to model one/eight mixture of Gaussian in

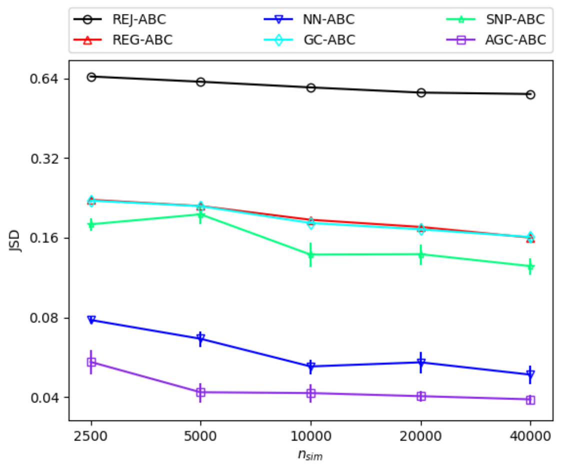

Figure 2: The JSD average across 15 runs in the GC toy problem under different simulation budgets.

the two phases respectively, with 50 tanh units in each layer. For AGC-ABC, the neural network, which is considered in model selection, has two hidden layers with 128 and 16 sigmoid units each. This is also the same network used in NN-ABC. All neural networks in the experiments are trained by Adam (Kingma and Ba, 2014) with its default setting.

5.2 Results

A toy problem The first problem we study is a 2D Gaussian copula model:

$$p(\mathbf{x}|\boldsymbol{\theta}) = c_{\text{CG}}\left(u_1, u_2; \begin{bmatrix} 1, & \theta_3 \\ \theta_3, & 1 \end{bmatrix}\right) \prod_{k=1}^{2} f_k(x_k; \theta_k)$$

$$f_1(x_1; \theta_1) = \text{Beta}(x_1; \theta_1, 2)$$

$$f_2(x_2; \theta_2) = \theta_2 \mathcal{N}(x_2; 1, 1) + (1 - \theta_2)\mathcal{N}(x_2; 4, 0.25)$$

where $u_k = F_k(x_k)$ is the value of the marginal CDF at x_k and $c_{\text{CG}}(\cdot)$ is the aforementioned Gaussian copula density. The parameters of interest are $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ and the true parameters are $\boldsymbol{\theta}^* = (6.0, 0.5, 0.6)$. We place a flat, independent prior on $\boldsymbol{\theta}$: $\theta_1 \sim \mathcal{U}(0.5, 12.5)$, $\theta_2 \sim \mathcal{U}(0, 1)$, $\theta_3 \sim \mathcal{U}(0.4, 0.8)$. The summary statistics $\mathbf{s}$ are taken to be the union of (a) the 20 equally-spaced marginal quantiles and (b) the correlation between $z_1 = \Phi^{-1}(u_1)$, $z_2 = \Phi^{-1}(u_2)$ of the generated data $\mathbf{x}^o = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(200)}\}$. In this problem, the true posterior is available analytically.

Figure 2 shows the JSD between the true and estimation posterior for the different methods (on log-scale, vertical lines indicate standard errors, each JSD is obtained by calculating the average of 15 runs for different observed data, the results shown in the figures below have the same setup). Expectedly given the model specification, AGC-ABC yields a lower JSD than the other methods for all simulation budgets. We further see that there is only a small difference in performance between AGC-ABC and NN-ABC, especially in large budget settings. A possible explanation is that the distribution of the residuals $p(\boldsymbol{\xi}|\mathbf{s})$ does only weakly depend on $\mathbf{s}$. To further investigate this, we compared the distribution of the residuals at $\mathbf{s}^o$ with the distribution $p_\epsilon(\boldsymbol{\xi})$ of the residuals within ϵ-balls of increasing radius around $\mathbf{s}^o$. Table 1 shows the JSD values between the two distributions for different values of ϵ.[1] It can be seen that the JSD between $p(\boldsymbol{\xi}|\mathbf{s}^o)$ and $p_\epsilon(\boldsymbol{\xi})$ increases as ϵ increases but the changes are not large. This is in line with the small gain of AGC-ABC over NN-ABC. The supplementary material further compares the contour plots of $p(\boldsymbol{\xi}|\mathbf{s}^o)$ and $p_\epsilon(\boldsymbol{\xi})$.

ϵ (quantile)	0.1%	1%	10%	25%
JSD$(p(\boldsymbol{\xi}\|\mathbf{s}^o), p_\epsilon(\boldsymbol{\xi}))$	0.040	0.041	0.047	0.055

Table 1: GC toy problem: The JSD values indicate a dependency of the residuals on $\mathbf{s}$.

The M/G/1 queueing problem. The M/G/1 queueing model is a real-world problem that describes the processing of incoming jobs in a single server system. Jobs in this model are served on a first-in-first-out (FIFO) basis. The whole data generating process in M/G/1 can be described as:

$$s_i \sim \mathcal{U}(\theta_1, \theta_1 + \theta_2)$$

$$v_i - v_{i-1} \sim \text{Exp}(\theta_3)$$

$$d_i - d_{i-1} = s_i + \max(0, v_i - d_{i-1})$$

where s_i, v_i, d_i are the service time, visiting time and finish time of job i respectively. The parameters of interest in this problem are $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ and the true parameter values are $\boldsymbol{\theta}^* = (1.0, 4.0, 0.2)$. Again, we place a flat, independent prior on $\boldsymbol{\theta}$: $\theta_1 \sim \mathcal{U}(0, 10)$, $\theta_2 \sim \mathcal{U}(2, 6)$, $\theta_3 \sim \mathcal{U}(0, 0.33)$. The observed data $\mathbf{x}^o$ in this problem are given by the time intervals $x_i = d_i - d_{i-1}$ between the finish time of two consecutive jobs. 200 such intervals are observed so that $\mathbf{x}^o = \{x_1, \ldots, x_{200}\}$. The summary statistics $\mathbf{s}^o$ are taken as the 16-equally spaced quantiles of $\mathbf{x}^o$. These statistics are further preprocessed to have zero mean and unit variance on each dimension.

Figure 3 shows the JSD of the different methods under various simulation budgets. We see that there is a larger gap between AGC-ABC and traditional regression ABC methods for small simulation budgets. This is because the underlying residual strongly depends on

[1] For $\epsilon = 50\%$, the JSD is 0.061.

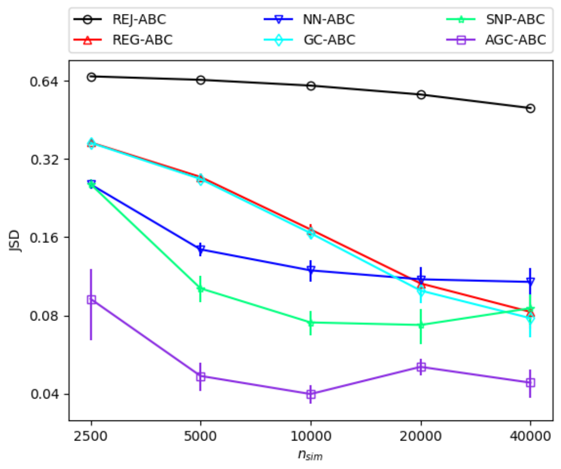

Figure 3: The JSD average across 15 runs in the M/G/1 problem under different simulation budgets.

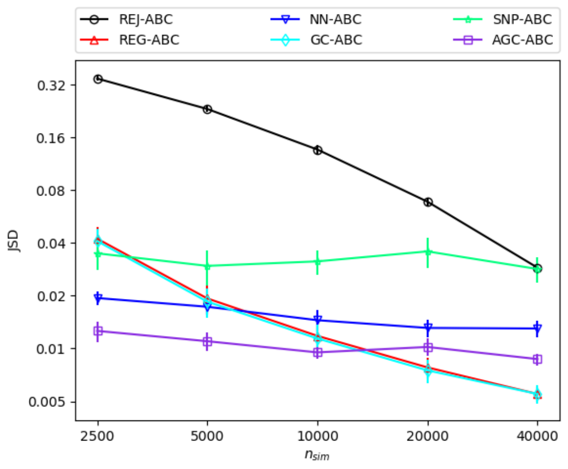

Figure 4: The JSD average across 15 runs in the MA(2) problem under different simulation budgets.

$\mathbf{s}$ in this problem; it is highly heterogeneous (see Table 2 and the contour plots in the supplementary material). This also points to why NN-ABC does here not perform as good as AGC-ABC — only modelling the dependency between the summary statistics and the scale (variance) of the residuals does not seem enough. SNP-ABC achieves very good performance compared to traditional ABC methods due to its heterogeneous modelling but it is still less accurate than AGC-ABC, which is natural since it requires more data to fit.

ϵ (quantile)	0.1%	1%	10%	25%
$\text{JSD}(p(\boldsymbol{\xi}\vert\mathbf{s}^o), p_\epsilon(\boldsymbol{\xi}))$	0.047	0.112	0.159	0.194

Table 2: M/G/1: JSD values between $p(\boldsymbol{\xi}|\mathbf{s}^o)$ and $p_\epsilon(\boldsymbol{\xi})$ indicate a dependency of the residuals on $\mathbf{s}$.

MA(2) time series problem The third problem we consider is the second order moving average model in time series analysis, known as MA(2). In this model, data are generated as follows

$$x^{(t)} = w^{(t)} + \theta_1 w^{(t-1)} + \theta_2 w^{(t-2)},$$

where $x^{(t)}$ is the observation at time t and $w^{(t)}$ is some unobservable standard normal noise. The parameters of interest are $\boldsymbol{\theta} = (\theta_1, \theta_2)$ and the true parameters are $\boldsymbol{\theta}^* = (0.6, 0.2)$. As in the previous problems, we place a flat and uniform prior on $\boldsymbol{\theta}$: $\theta_1 \sim \mathcal{U}(0,1)$, $\theta_2 \sim \mathcal{U}(0,1)$. Following the ABC literature (Jiang et al., 2017; Marin et al., 2012; Lintusaari et al., 2018), we adopt the autocovariance with lag 1 and lag 2 as the summary statistics. These statistics are computed from a time series $\mathbf{x}^o = (x^{(1)}, \ldots, x^{(200)})$ of length 200. The statistics are further preprocessed to have zero mean and unit variance for each dimension.

Figure 4 reports the JSD of the different methods under various simulation budgets. While AGC-ABC performs best for small budgets, the simpler REG-ABC method performs better in the larger budget case. This might be due to Gaussian copula being inadequate for the modelling of the residuals. However, the difference between AGC-ABC and REG-ABC is not large as evidenced by the contour plots in the supplementary materials. We also see that the 8-mixture of Gaussian construction in SNP-ABC might be not powerful enough for this problem, so that it incurs a bias which is not eliminated as the simulation budget increases.

ϵ (quantile)	0.1%	1%	10%	25%
$\text{JSD}(p(\boldsymbol{\xi}\vert\mathbf{s}^o), p_\epsilon(\boldsymbol{\xi}))$	0.004	0.004	0.006	0.010

Table 3: MA(2): JSD values between $p(\boldsymbol{\xi}|\mathbf{s}^o)$ and $p_\epsilon(\boldsymbol{\xi})$ indicate a dependency of the residuals on $\mathbf{s}$.

Lotka-Volterra problem The last problem we consider is a stochastic dynamical system in biology that describes predator-prey dynamics. There are four possible events in the system: (a) a predator being born, (b) a predator dying, (c) a prey being born, (d) a prey being eaten by the predator. The events can be described probabilistically as

$$\begin{aligned}
\mathbb{P}(X \to X+1|\boldsymbol{\theta}) &\propto \exp(\theta_1)\cdot XY,\\
\mathbb{P}(X \to X-1|\boldsymbol{\theta}) &\propto \exp(\theta_2)\cdot X,\\
\mathbb{P}(Y \to Y+1|\boldsymbol{\theta}) &\propto \exp(\theta_3)\cdot Y,\\
\mathbb{P}(Y \to Y-1|\boldsymbol{\theta}) &\propto \exp(\theta_1)\cdot XY,
\end{aligned}$$

where X, Y are the numbers of predators and preys respectively. For a more rigorous formulation see e.g.

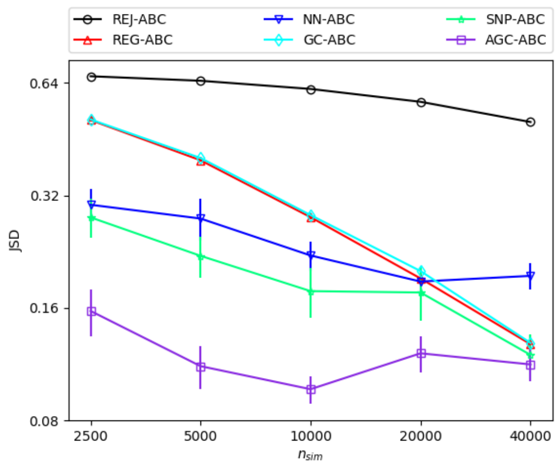

Figure 5: The JSD average across 15 runs in the LV problem under different simulation budgets.

(Fearnhead and Prangle, 2012). The parameters of interest are $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$ and the true parameters are $\boldsymbol{\theta}^* = (\log(0.01), \log(0.5), \log(0.1))$. The initial sizes of the populations are $(X^0, Y^0) = (50, 100)$. While the likelihood is intractable, sampling from the model is possible (Gillespie, 1977). We place a flat and independent prior on $\boldsymbol{\theta}$: $\theta_1 \sim \mathcal{U}(-5, -1)$, $\theta_2 \sim \mathcal{U}(-1, 1)$, $\theta_3 \sim \mathcal{U}(-1, 1)$. We simulate the system for a total of 8 time units and record the values of X and Y after every 0.2 time units. This yields two time series of length 41 being our observed data $\mathbf{x}^o$. The summary statistics are taken as the records between every two consecutive time points.

Figure 5 shows the JSD of the different methods. The results show that AGC-ABC offers a better efficiency-accuracy trade-off than the other methods, with a clear performance gain for small simulation budgets. This is due to the fact that the underlying distribution $p(\boldsymbol{\xi}|\mathbf{s})$ is highly heterogeneous (see Table 4 and the the contour plots of $p(\boldsymbol{\xi}|\mathbf{s})$ in the supplementary material).

ϵ (quantile)	0.1%	1%	10%	25%
JSD$(p(\boldsymbol{\xi}|\mathbf{s}^o), p_\epsilon(\boldsymbol{\xi}))$	0.089	0.108	0.149	0.194

Table 4: LV: JSD values between $p(\boldsymbol{\xi}|\mathbf{s}^o)$ and $p_\epsilon(\boldsymbol{\xi})$ indicate a dependency of the residuals on $\mathbf{s}$.

6 Conclusion

We considered the problem of posterior density estimation when the likelihood is intractable but sampling from the model is possible. We proposed a new method for approximate Bayesian computation (ABC) that combines the basic ideas from two different lines of ABC research, namely regression ABC and sequential ABC. We found that the resulting algorithm strikes a good trade-off between accuracy and computational cost, being particularly effective in the regime of smaller simulation budgets.

The motivation behind the proposed algorithm was the homogeneity assumption on the residuals that is required for regression ABC to work well. The proposed method takes a sequential approach by first generating training data so that the homogeneity assumption is better satisfied, and then models the data with the aid of a Gaussian copula and existing techniques from regression ABC. The method — adaptive Gaussian copula ABC — can thus either be viewed as an adaptive version of classical regression ABC methods (Beaumont et al., 2002; Blum and François, 2010), or a computationally cheaper version of recent sequential neural posterior approaches (Papamakarios and Murray, 2016; Lueckmann et al., 2017).

While Gaussian copula are powerful they are not silver bullets. Extending the Gaussian copula model to other statistical models such as mixture of Gaussian copula (Fan et al., 2013; Bilgrau et al., 2016) or vine copulas (Bedford and Cooke, 2002; Aas and Berg, 2009) may be a future research direction worth exploring.

Acknowledgement

We would like to thank the anonymous reviewers for their insightful and fruitful suggestions.

References

Aas, K. and Berg, D. (2009). Models for construction of multivariate dependence–a comparison study. *The European Journal of Finance*, 15(7-8):639–659.

Beaumont, M. A. (2010). Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41(1):379–406.

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

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

Bedford, T. and Cooke, R. M. (2002). Vines: A new graphical model for dependent random variables. *Annals of Statistics*, pages 1031–1068.

Bilgrau, A. E., Eriksen, P. S., Rasmussen, J. G., Johnsen, H. E., Dybkaer, K., and Boegsted, M.

(2016). Gmcm: Unsupervised clustering and meta-analysis using gaussian mixture copula models. *Journal of Statistical Software*, 70(i02).

Blum, M. G. (2010). Approximate Bayesian computation: a nonparametric perspective. *Journal of the American Statistical Association*, 105(491):1178–1187.

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

Corander, J., Fraser, C., Gutmann, M., Arnold, B., Hanage, W., Bentley, S., Lipsitch, M., and Croucher, N. (2017). Frequency-dependent selection in vaccine-associated pneumococcal population dynamics. *Nature Ecology and Evolution*, 1:1950–1960.

Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227.

Fan, Y., Nott, D. J., and Sisson, S. A. (2013). Approximate Bayesian computation via regression density estimation. *Stat*, 2(1):34–48.

Fearnhead, P. and Prangle, D. (2012). Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474.

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361.

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

Jiang, B., Wu, T.-Y., Zheng, C., and Wong, W. H. (2017). Learning summary statistic for approximate Bayesian computation via deep neural network. *Statistica Sinica*, 27:1595–1618.

Karabatsos, G. (2017). On bayesian testing of additive conjoint measurement axioms using synthetic likelihood. *Psychometrika*, pages 1–12.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, J., Nott, D. J., Fan, Y., and Sisson, S. A. (2017). Extending approximate Bayesian computation methods to high dimensions via a gaussian copula model. *Computational Statistics and Data Analysis*, 106:77–89.

Lintusaari, J., Gutmann, M., Dutta, R., Kaski, S., and Corander, J. (2017). Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, 66(1):e66–e82.

Lintusaari, J., Vuollekoski, H., Kangasrääsiö, A., Skytén, K., Järvenpää, M., Gutmann, M., Vehtari, A., Corander, J., and Kaski, S. (2018). ELFI: Engine for likelihood-free inference. *Journal of Machine Learning Research*, 19(16).

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

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012). Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180.

Marjoram, P., Molitor, J., Plagnol, V., and Tavaré, S. (2003). Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328.

Papamakarios, G. and Murray, I. (2016). Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1028–1036.

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

Scott, D. W. (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Sisson, S., Fan, Y., and Beaumont, M. (2018). *Handbook of Approximate Bayesian Computation.*, chapter Overview of Approximate Bayesian Computation. Chapman and Hall/CRC Press.

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

Weyant, A., Schafer, C., and Wood-Vasey, W. M. (2013). Likelihood-free cosmological inference with type ia supernovae: approximate Bayesian computation for a complete treatment of uncertainty. *The Astrophysical Journal*, 764(2).

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