Nesting Probabilistic Programs

Tom Rainforth
Department of Statistics
University of Oxford
rainforth@stats.ox.ac.uk

Abstract

We formalize the notion of nesting probabilistic programming queries and investigate the resulting statistical implications. We demonstrate that while query nesting allows the definition of models which could not otherwise be expressed, such as those involving agents reasoning about other agents, existing systems take approaches which lead to inconsistent estimates. We show how to correct this by delineating possible ways one might want to nest queries and asserting the respective conditions required for convergence. We further introduce a new *online* nested Monte Carlo estimator that makes it substantially easier to ensure these conditions are met, thereby providing a simple framework for designing statistically correct inference engines. We prove the correctness of this online estimator and show that, when using the recommended setup, its asymptotic variance is always better than that of the equivalent fixed estimator, while its bias is always within a factor of two.

1 INTRODUCTION

Probabilistic programming systems (PPSs) allow probabilistic models to be represented in the form of a generative model and statements for conditioning on data (Goodman et al., 2008; Gordon et al., 2014). Informally, one can think of the generative model as the definition of a prior, the conditioning statements as the definition of a likelihood, and the output of the program as samples from a posterior distribution. Their core philosophy is to decouple model specification and inference, the former corresponding to the user-specified program code and the latter to an inference engine capable of operating on arbitrary programs. Removing the need for users to write inference algorithms significantly reduces the burden of developing new models and makes effective statistical methods accessible to non-experts.

Some, so-called universal, systems (Goodman et al., 2008; Goodman and Stuhlmüller, 2014; Mansinghka et al., 2014; Wood et al., 2014) further allow the definition of models that would be hard, or even impossible, to convey using conventional frameworks such as graphical models. One enticing manner they do this is by allowing arbitrary nesting of models, known in the probabilistic programming literature as queries (Goodman et al., 2008), such that it is easy to define and run problems that fall outside the standard inference framework (Goodman et al., 2008; Mantadelis and Janssens, 2011; Stuhlmüller and Goodman, 2014; Le et al., 2016). This allows the definition of models that could not be encoded without nesting, such as experimental design problems (Ouyang et al., 2016) and various models for theory-of-mind (Stuhlmüller and Goodman, 2014). In particular, models that involve agents reasoning about other agents require, in general, some form of nesting. For example, one might use such nesting to model a poker player reasoning about another player as shown in Section 3.1. As machine learning increasingly starts to try and tackle problem domains that require interaction with humans or other external systems, such as the need for self-driving cars to account for the behavior of pedestrians, we believe that such nested problems are likely to become increasingly common and that PPSs will form a powerful tool for encoding them.

However, previous work has, in general, implicitly, and incorrectly, assumed that the convergence results from standard inference schemes carry over directly to the nested setting. In truth, inference for nested queries falls outside the scope of conventional proofs and so additional work is required to prove the consistency of PPS inference engines for nested queries. Such problems constitute special cases of ***nested estimation***. In particular, the use of Monte Carlo (MC) methods by most PPSs mean they form particular instances of ***nested Monte Carlo*** (NMC) estimation (Hong and Juneja, 2009). Recent work (Rainforth et al., 2016a, 2018; Fort et al., 2017) has demonstrated that NMC is consistent for a general class of models, but also that it entails a convergence rate in the total computational cost which decreases exponentially with the depth of the nesting. Furthermore, additional assumptions are required to achieve this convergence, most noticeably that, except in a few special cases, one needs to drive not only the total number of samples used to infinity, but also the number of samples used at each layer of the estimator, a requirement generally flaunted by existing PPSs.

The aim of this work is to formalize the notion of query nesting and use these recent NMC results to investigate the statistical correctness of the resulting procedures carried out by PPS inference engines. To do this, we postulate that there are three distinct ways one might nest one query within another: sampling from the conditional distribution of another query (which we refer to as ***nested inference***), factoring the trace probability of one query with the partition function estimate of another (which we refer to as ***nested conditioning***), and using expectation estimates calculated using one query as ***first class variables*** in another. We use the aforementioned NMC results to assess the relative correctness of each of these categories of nesting. In the interest of exposition, we will mostly focus on the PPS *Anglican* (Tolpin et al., 2016; Wood et al., 2014) (and also occasionally Church (Goodman et al., 2008)) as a basis for our discussion, but note that our results apply more generally. For example, our nested inference case covers the problem of sampling from cut distributions in OpenBugs (Plummer, 2015).

We find that nested inference is statistically challenging and incorrectly handled by existing systems, while nested conditioning is statistically straightforward and done correctly. Using estimates as variables turns out to be exactly equivalent to generic NMC estimation and must thus be dealt with on a case-by-case basis. Consequently, we will focus more on nested inference than the other cases.

To assist in the development of consistent approaches, we further introduce a new *online* NMC (ONMC) scheme that obviates the need to revisit previous samples when refining estimates, thereby simplifying the process of writing consistent online nested estimation schemes, as required by most PPSs. We show that ONMC's convergence rate only varies by a small constant factor relative to conventional NMC: given some weak assumptions and the use of recommended parameter settings, its asymptotic variance is always better than the equivalent NMC estimator with matched total sample budget, while its asymptotic bias is always within a factor of two.

2 BACKGROUND

2.1 NESTED MONTE CARLO

We start by providing a brief introduction to NMC, using similar notation to that of Rainforth et al. (2018). Conventional MC estimation approximates an intractable expectation γ_0 of a function λ using

$$\gamma_0 = \mathbb{E}\left[\lambda(y^{(0)})\right] \approx I_0 = \frac{1}{N_0}\sum_{n=1}^{N_0} \lambda(y_n^{(0)}) \qquad (1)$$

where $y_n^{(0)} \overset{i.i.d.}{\sim} p(y^{(0)})$, resulting in a mean squared error (MSE) that decreases at a rate $O(1/N_0)$. For nested estimation problems, $\lambda(y^{(0)})$ is itself intractable, corresponding to a nonlinear mapping of a (nested) estimation. Thus in the single nesting case, $\lambda(y^{(0)}) = f_0\left(y^{(0)}, \mathbb{E}\left[f_1\left(y^{(0)}, y^{(1)}\right)\middle|y^{(0)}\right]\right)$ giving

$$\gamma_0 = \mathbb{E}\left[f_0\left(y^{(0)}, \mathbb{E}\left[f_1\left(y^{(0)}, y^{(1)}\right)\middle|y^{(0)}\right]\right)\right]$$
$$\approx I_0 = \frac{1}{N_0}\sum_{n=1}^{N_0} f_0\left(y_n^{(0)}, \frac{1}{N_1}\sum_{m=1}^{N_1} f_1\left(y_n^{(0)}, y_{n,m}^{(1)}\right)\right)$$

where each $y_{n,m}^{(1)} \sim p(y^{(1)}|y_n^{(0)})$ is drawn independently and I_0 is now a NMC estimate using $T = N_0 N_1$ samples.

More generally, one may have multiple layers of nesting. To notate this, we first presume some fixed integral depth $D \geq 0$ (with $D = 0$ corresponding to conventional estimation), and real-valued functions $f_0, \ldots, f_D$. We then recursively define

$$\gamma_D\left(y^{(0:D-1)}\right) = \mathbb{E}\left[f_D\left(y^{(0:D)}\right)\middle|y^{(0:D-1)}\right], \quad \text{and}$$
$$\gamma_k(y^{(0:k-1)}) = \mathbb{E}\left[f_k\left(y^{(0:k)}, \gamma_{k+1}\left(y^{(0:k)}\right)\right)\middle|y^{(0:k-1)}\right]$$

for $0 \leq k < D$. Our goal is to estimate $\gamma_0 = \mathbb{E}\left[f_0\left(y^{(0)}, \gamma_1\left(y^{(0)}\right)\right)\right]$, for which the NMC estimate is I_0 defined recursively using

$$I_D\left(y^{(0:D-1)}\right) = \frac{1}{N_D}\sum_{n_D=1}^{N_D} f_D\left(y^{(0:D-1)}, y_{n_D}^{(D)}\right) \quad \text{and}$$
$$I_k\left(y^{(0:k-1)}\right) \qquad (2)$$
$$= \frac{1}{N_k}\sum_{n_k=1}^{N_k} f_k\left(y^{(0:k-1)}, y_{n_k}^{(k)}, I_{k+1}\left(y^{(0:k-1)}, y_{n_k}^{(k)}\right)\right)$$

for $0 \leq k < D$, where each $y_n^{(k)} \sim p\left(y^{(k)}|y^{(0:k-1)}\right)$ is drawn independently. Note that there are multiple values of $y^{(k)}$ for each associated $y^{(0:k-1)}$ and that $I_k\left(y^{(0:k-1)}\right)$ is still a random variable given $y^{(0:k-1)}$.

As shown by (Rainforth et al., 2018, Theorem 3), if each f_k is continuously differentiable and

$$\varsigma_k^2 = \mathbb{E}\left[\left(f_k\left(y^{(0:k)}, \gamma_{k+1}\left(y^{(0:k)}\right)\right) - \gamma_k\left(y^{(0:k-1)}\right)\right)^2\right]$$

$< \infty \; \forall k \in 0, \ldots, D$, then the MSE converges at rate

$$\mathbb{E}\left[\left(I_0 - \gamma_0\right)^2\right] \leq \frac{\varsigma_0^2}{N_0} +$$
$$\left(\frac{C_0 \varsigma_1^2}{2N_1} + \sum_{k=0}^{D-2}\left(\prod_{d=0}^{k} K_d\right)\frac{C_{k+1}\varsigma_{k+2}^2}{2N_{k+2}}\right)^2 + O(\epsilon) \qquad (3)$$

where K_k and C_k are respectively bounds on the magnitude of the first and second derivatives of f_k, and $O(\epsilon)$ represents asymptotically dominated terms – a convention we will use throughout. Note that the dominant terms in the bound correspond respectively to the variance and the bias squared. Theorem 2 of Rainforth et al. (2018) further shows that the continuously differentiable assumption must hold almost surely, rather than absolutely, for convergence more generally, such that functions with measure-zero discontinuities still converge in general.

We see from (3) that if any of the N_k remain fixed, there is a minimum error that can be achieved: convergence requires each $N_k \to \infty$. As we will later show, many of the shortfalls in dealing with nested queries by existing PPSs revolve around implicitly fixing $N_k \, \forall k \geq 1$.

For a given total sample budget $T = N_0 N_1 \dots N_D$, the bound is tightest when $\sqrt{N_0} \propto N_1 \propto \cdots \propto N_D$ giving a convergence rate of $O(1/T^{\frac{2}{D+2}})$. The intuition behind this potentially surprising optimum setting is that the variance is mostly dictated by N_0 and bias by the other N_k. We see that the convergence rate diminishes exponentially with D. However, this optimal setting of the N_k still gives a substantially faster rate than the $O(1/T^{\frac{1}{D+1}})$ from naïvely setting $N_0 \propto N_1 \propto \cdots \propto N_D$.

2.2 THE ANGLICAN PPS

Anglican is a universal probabilistic programming language integrated into *Clojure* (Hickey, 2008), a dialect of Lisp. There are two important ideas to understand for reading Clojure: almost everything is a function and parentheses cause evaluation. For example, $a + b$ is coded as `(+ a b)` where `+` is a function taking two arguments and the parentheses cause the function to evaluate.

Anglican inherits most of the syntax of Clojure, but extends it with the key special forms `sample` and `observe` (Wood et al., 2014; Tolpin et al., 2015, 2016), between which the distribution of the query is defined. Informally, `sample` specifies terms in the prior and `observe` terms in the likelihood. More precisely, `sample` is used to make random draws from a provided distribution and `observe` is used to apply conditioning, factoring the probability density of a program trace by a provided density evaluated at an "observed" point.

The syntax of `sample` is to take a *distribution object* as its only input and return a sample. `observe` instead takes a distribution object and an observation and returns `nil`, while changing the program trace probability in Anglican's back-end. Anglican provides a number of *elementary random procedures*, i.e. distribution object constructors for common sampling distributions, but also allows users to define their own distribution object constructors using the `defdist` macro. Distribution objects are generated by calling a class constructor with the required parameters, e.g. `(normal 0 1)`.

Anglican queries are written using the macro `defquery`. This allows users to define a model using a mixture of `sample` and `observe` statements and deterministic code, and bind that model to a variable. As a simple example,

```
(defquery my-query [data]
 (let [μ (sample (normal 0 1))
       σ (sample (gamma 2 2))
       lik (normal μ σ)]
  (map (fn [obs] (observe lik obs)) data)
  [μ σ]))
```

corresponds to a model where we are trying to infer the mean and standard deviation of a Gaussian given some data. The syntax of `defquery` is `(defquery name [args] body)` such that we are binding the query to `my-query` here. The query starts by sampling $\mu \sim \mathcal{N}(0,1)$ and $\sigma \sim \Gamma(2,2)$, before constructing a distribution object `lik` to use for the observations. It then maps over each datapoint and observes it under the distribution `lik`. After the observations are made, μ and σ are returned from the variable-binding `let` block and then by proxy the query itself. Denoting the data as $y_{1:S}$ this particular query defines the joint distribution

$$p(\mu, \sigma, y_{1:S}) = \mathcal{N}(\mu; 0, 1)\, \Gamma(\sigma; 2, 2) \prod\nolimits_{s=1}^{S} \mathcal{N}(y_s; \mu, \sigma).$$

Inference on a query is performed using the macro `doquery`, which produces a lazy infinite sequence of approximate samples from the conditional distribution and, for appropriate inference algorithms, an estimate of the partition function. Its calling syntax is `(doquery inf-alg model inputs & options)`.

Key to our purposes is Anglican's ability to *nest* queries within one another. In particular, the special form `conditional` takes a query and returns a distribution object constructor, the outputs of which ostensibly corresponds to the conditional distribution defined by the query, with the inputs to the query becoming its parameters. However, as we will show in the next section, the true behavior of `conditional` deviates from this, thereby leading to inconsistent nested inference schemes.

3 NESTED INFERENCE

One of the clearest ways one might want to nest queries is by sampling from the conditional distribution of one query inside another. A number of examples of this are provided for Church in (Stuhlmüller and Goodman, 2014).[1] Such *nested inference* problems fall under a more general framework of inference for so-called doubly (or multiply) intractable distributions (Murray et al., 2006). The key feature of these problems is that they include terms with unknown, *parameter dependent*, normalization constants. For nested probabilistic programming queries, this manifests through *conditional normalization*.

[1] Though their nesting happens within the conditioning predicate, Church's semantics means they constitute nested inference.

Consider the following unnested model using the Anglican function declaration defm

```
(defm inner [y D]
 (let [z (sample (gamma y 1))]
  (observe (normal y z) D)
  z))

(defquery outer [D]
 (let [y (sample (beta 2 3))
       z (inner y D)]
  (* y z)))
```

Here inner is simply an Anglican function: it takes in inputs y and D, effects the trace probability through its observe statement, and returns the random variable z as output. The unnormalized distribution for this model is thus straightforwardly given by

$$\begin{aligned}\pi_u(y, z, D) &= p(y)p(z|y)p(D|y, z)\\ &= \text{BETA}(y; 2, 3)\,\Gamma(z; y, 1)\,\mathcal{N}(D; y, z^2),\end{aligned}$$

for which we can use conventional inference schemes.

We can convert this model to a nested inference problem by using defquery and conditional as follows

```
(defquery inner [y D]
 (let [z (sample (gamma y 1))]
  (observe (normal y z) D)
  z))

(defquery outer [D]
 (let [y (sample (beta 2 3))
       dist (conditional inner)
       z (sample (dist y D))]
  (* y z)))
```

This is now a nested query: a separate inference procedure is invoked for each call of (sample (dist y D)), returning an approximate sample from the conditional distribution defined by inner when input with the current values of y and D. Mathematically, conditional applies a conditional normalization. Specifically, the component of π_u from the previous example corresponding to inner was $p(z|y)p(D|y, z)$ and conditional locally normalizes this to the probability distribution $p(z|D, y)$. The distribution now defined by outer is thus given by

$$\begin{aligned}\pi_n(y, z, D) = p(y)p(z|y, D) &= \frac{p(y)p(z|y)p(D|y, z)}{\int p(z|y)p(D|y, z)dz}\\ &= p(y)\frac{p(z|y)p(D|y, z)}{p(D|y)} \neq \pi_u(z, y, D).\end{aligned}$$

Critically, the partial normalization constant $p(D|y)$ depends on y and so the conditional distribution is doubly intractable: we cannot evaluate $\pi_n(y, z, D)$ exactly.

Another way of looking at this is that wrapping inner in conditional has "protected" y from the conditioning in inner (noting $\pi_u(y, z, D) \propto p(y|D)p(z|y, D)$), such that its observe statement only affects the probability of z given y and not the marginal probability of y. This is why, when there is only a single layer of nesting, nested inference is equivalent to the notion of sampling from "cut distributions" (Plummer, 2015), whereby the sampling of certain subsets of the variables in a model are made with factors of the overall likelihood omitted.

It is important to note that if we had observed the *output* of the inner query, rather than sampling from it, this would still constitute a nested inference problem. The key to the nesting is the conditional normalization applied by conditional, not the exact usage of the generated distribution object dist. However, as discussed in Appendix B, actually observing a nested query requires numerous additional computational issues to be overcome, which are beyond the scope of this paper. We thus focus on the nested sampling scenario.

3.1 MOTIVATING EXAMPLE

Before jumping into a full formalization of nested inference, we first consider the motivating example of modeling a poker player who reasons about another player. Here each player has access to information the other does not, namely the cards in their hand, and they must perform their own inference to deal with the resulting uncertainty.

Imagine that the first player is deciding whether or not to bet. She could naïvely just make this decision based on the strength of her hand, but more advanced play requires her to reason about actions the other player might take given her own action, e.g. by considering whether a bluff is likely to be successful. She can carry out such reasoning by constructing a model for the other player to try and predict their action given her action and their hand. Again this nested model could just simply be based on a naïve simulation, but we can refine it by adding another layer of meta-reasoning: the other player will themselves try to infer the first player's hand to inform their own decision.

These layers of meta-reasoning create a nesting: for the first player to choose an action, they must run multiple simulations for what the other player will do given that action and their hand, each of which requires inference to be carried out. Here adding more levels of meta-reasoning can produce smarter models, but also requires additional layers of nesting. We expand on this example to give a concrete nested inference problem in Appendix E.

3.2 FORMALIZATION

To formalize the nested inference problem more generally, let y and x denote all the random variables of an outer query that are respectively passed or not to the inner query. Further, let z denote all random variables generated in the inner query – for simplicity, we will assume, without loss of generality, that these are all returned to the outer query, but that some may not be used. The unnormalized density for the outer query can now be written in the form

$$\pi_o(x, y, z) = \psi(x, y, z)p_i(z|y) \tag{4}$$

where $p_i(z|y)$ is the normalized density of the outputs of the inner query and $\psi(x, y, z)$ encapsulates all other terms influencing the trace probability of the outer query. Now the inner query defines an unnormalized density $\pi_i(y, z)$ that can be evaluated pointwise and we have

$$p_i(z|y) = \frac{\pi_i(y, z)}{\int \pi_i(y, z')dz'} \quad \text{giving} \tag{5}$$

$$p_o(x, y, z) \propto \pi_o(x, y, z) = \frac{\psi(x, y, z)\pi_i(y, z)}{\int \pi_i(y, z')dz'} \tag{6}$$

where $p_o(x, y, z)$ is our target distribution, for which we can directly evaluate the numerator, but the denominator is intractable and must be evaluated separately for each possible value of y. Our previous example is achieved by fixing $\psi(x, y, z) = p(y)$ and $\pi_i(y, z) = p(z|y)p(D|y, z)$. We can further straightforwardly extend to the multiple layers of nesting setting by recursively defining $\pi_i(y, z)$ in the same way as $\pi_o(x, y, z)$.

3.3 RELATIONSHIP TO NESTED ESTIMATION

To relate the nested inference problem back to the nested estimation formulation from Section 2.1, we consider using a proposal $q(x, y, z) = q(x, y)q(z|y)$ to calculate the expectation of some arbitrary function $g(x, y, z)$ under $p_o(x, y, z)$ as per self-normalized importance sampling

$$\mathbb{E}_{p_o(x,y,z)}\left[g(x, y, z)\right] = \frac{\mathbb{E}_{q(x,y,z)}\left[\frac{g(x,y,z)\pi_o(x,y,z)}{q(x,y,z)}\right]}{\mathbb{E}_{q(x,y,z)}\left[\frac{\pi_o(x,y,z)}{q(x,y,z)}\right]}$$

$$= \frac{\mathbb{E}_{q(x,y,z)}\left[\dfrac{g(x, y, z)\psi(x, y, z)\pi_i(y, z)}{q(x, y, z)\mathbb{E}_{z'\sim q(z|y)}\left[\pi_i(y, z')/q(z'|y)\right]}\right]}{\mathbb{E}_{q(x,y,z)}\left[\dfrac{\psi(x, y, z)\pi_i(y, z)}{q(x, y, z)\mathbb{E}_{z'\sim q(z|y)}\left[\pi_i(y, z')/q(z'|y)\right]}\right]}. \tag{7}$$

Here both the denominator and numerator are nested expectations with a nonlinearity coming from the fact that we are using the reciprocal of an expectation. A similar reformulation could also be applied in cases with multiple layers of nesting, i.e. where `inner` itself makes use of another query. The formalization can also be directly extended to the sequential MC (SMC) setting by invoking extended space arguments (Andrieu et al., 2010).

Typically $g(x, y, z)$ is not known upfront and we instead return an empirical measure from the program in the form of weighted samples which can later be used to estimate an expectation. That is, if we sample $(x_n, y_n) \sim q(x, y)$ and $z_{n,m} \sim q(z|y_n)$ and return all samples $(x_n, y_n, z_{n,m})$ (such that each (x_n, y_n) is duplicated N_1 times in the sample set) then our unnormalized weights are given by

$$w_{n,m} = \frac{\psi(x_n, y_n, z_{n,m})\pi_i(y_n, z_{n,m})}{q(x_n, y_n, z_{n,m})\frac{1}{N_1}\sum_{\ell=1}^{N_1}\frac{\pi_i(y_n, z_{n,\ell})}{q(z_{n,\ell}|y_n)}}. \tag{8}$$

This, in turn, gives us the empirical measure

$$\hat{p}(\cdot) = \frac{\sum_{n=1}^{N_0}\sum_{m=1}^{N_1} w_{n,m}\delta_{(x_n,y_n,z_{n,m})}(\cdot)}{\sum_{n=1}^{N_0}\sum_{m=1}^{N_1} w_{n,m}} \tag{9}$$

where $\delta_{(x_n,y_n,z_{n,m})}(\cdot)$ is a delta function centered on $(x_n, y_n, z_{n,m})$. By definition, the convergence of this empirical measure to the target requires that expectation estimates calculated using it converge in probability for *any* integrable $g(x, y, z)$ (presuming our proposal is valid). We thus see that the convergence of the ratio of nested expectations in (7) for any arbitrary $g(x, y, z)$, is equivalent to the produced samples converging to the distribution defined by the program. Informally, the NMC results then tell us this will happen in the limit $N_0, N_1 \to \infty$ provided that $\int \pi_i(y, z)dz$ is strictly positive for all possible y (as otherwise the problem becomes ill-defined). More formally we have the following result. Its proof, along with all others, is given in Appendix A.

Theorem 1. *Let $g(x, y, z)$ be an integrable function, let $\gamma_0 = \mathbb{E}_{p_o(x,y,z)}[g(x, y, z)]$, and let I_0 be a self-normalized MC estimate for γ_0 calculated using $\hat{p}(\cdot)$ as per* (9)*. Assuming that $q(x, y, z)$ forms a valid importance sampling proposal distribution for $p_o(x, y, z)$, then*

$$\mathbb{E}\left[(I_0 - \gamma_0)^2\right] = \frac{\sigma^2}{N_0} + \frac{\delta^2}{N_1^2} + O(\epsilon) \tag{10}$$

where σ and δ are constants derived in the proof and, as before, $O(\epsilon)$ represents asymptotically dominated terms.

Note that rather than simply being a bound, this result is an equality and thus provides the exact asymptotic rate. Using the arguments of (Rainforth et al., 2018, Theorem 3), it can be straightforwardly extended to cases of multiple nesting (giving a rate analogous to (3)), though characterizing σ and δ becomes more challenging.

3.4 CONVERGENCE REQUIREMENTS

We have demonstrated that the problem of nested inference is a particular case of nested estimation. This problem equivalence will hold whether we elect to use the aforementioned nested importance sampling based approach or not, while we see that our finite sample estimates must be biased for non-trivial g by the convexity of f_0 and Theorem 4 of Rainforth et al. (2018). Presuming we cannot produce exact samples from the inner query and that the set of possible inputs to the inner query is not finite (these are respectively considered in Appendix D and Appendix C), we thus see that there is no "silver bullet" that can reduce the problem to a standard estimation.

We now ask, what behavior do we need for Anglican's **`conditional`**, and nested inference more generally, to ensure convergence? At a high level, the NMC results show us that we need the computational budget of each call of a nested query to become arbitrarily large, such that we use an infinite number of samples at each layer of the estimator: we require each $N_k \to \infty$.

We have formally demonstrated convergence when this requirement is satisfied and the previously introduced nested importance sampling approach is used. Another possible approach would be to, instead of drawing samples to estimate (7) directly, importance sample N_1 times for each call of the inner query and then return a single sample from these, drawn in proportion to the inner query importance weights. We can think of this as drawing the same raw samples, but then constructing the estimator as

$$\hat{p}^*(\cdot) = \frac{\sum_{n=1}^{N_0} w_n^* \delta_{(x_n, y_n, z_{n,m^*(n)})}(\cdot)}{\sum_{n=1}^{N_0} w_n^*} \tag{11}$$

$$\text{where} \quad w_n^* = \frac{\psi(x_n, y_n, z_{n,m^*(n)})}{q(x_n, y_n)} \quad \text{and} \tag{12}$$

$$m^*(n) \sim \text{DISCRETE}\left(\frac{\pi_i(y_n, z_{n,m})/q(z_{n,m}|y_n)}{\sum_{\ell=1}^{N_1} \pi_i(y_n, z_{n,\ell})/q(z_{n,\ell}|y_n)}\right)$$

As demonstrated formally in Appendix A, this approach also converges. However, if we Rao Blackwellize (Casella and Robert, 1996) the sampling of $m^*(n)$, we find that this recovers (9). Consequently, this is a strictly inferior estimator (it has an increased variance relative to (9)). Nonetheless, it may often be a convenient setup from the perspective of the PPS semantics and it will typically have substantially reduced memory requirements: we need only store the single returned sample from the inner query to construct our empirical measure, rather than all of the samples generated within the inner query.

Though one can use the results of Fort et al. (2017) to show the correctness of instead using an MCMC estimator for the outer query, the correctness of using MCMC methods for the inner queries is not explicitly covered by existing results. Here we find that we need to start a new Markov chain for each call of the inner query because each value of y defines a different local inference problem. One would intuitively expect the NMC results to carry over – as $N_1 \to \infty$ all the inner queries will run their Markov chains for an infinitely long time, thereby in principle returning exact samples – but we leave formal proof of this case to future work. We note that such an approach effectively equates to what is referred to as *multiple imputation* by Plummer (2015).

3.5 SHORTFALLS OF EXISTING SYSTEMS

Using the empirical measure (9) provides one possible manner of producing a consistent estimate of our target by taking $N_0, N_1 \to \infty$ and so we can use this as a gold-standard reference approach (with a large value of N_1) to assess whether Anglican returns samples for the correct target distribution. To this end, we ran Anglican's importance sampling inference engine on the simple model introduced earlier and compared its output to the reference approach using $N_0 = 5 \times 10^6$ and $N_1 = 10^3$. As shown in Figure 1, the samples produced by Anglican are substantially different to the reference code, demonstrating that the outputs do not match their semantically intended distribution. For reference, we also considered the distribution induced by the aforementioned unnested model and a naïve estimation scheme where a sample budget of $N_1 = 1$ is used for each call to `inner`, effectively corresponding to ignoring the **`observe`** statement by directly returning the first draw of z.

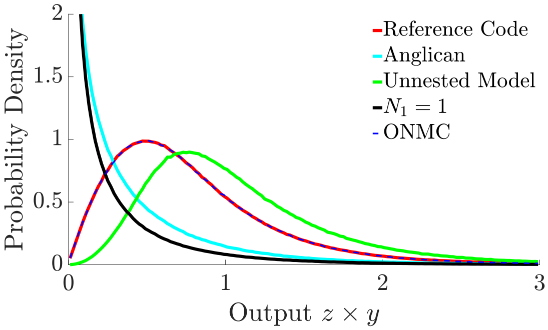

Figure 1: Empirical densities produced by running the nested Anglican queries given in the text, a reference NMC estimate, the unnested model, a naïve estimation scheme where $N_1 = 1$, and the ONMC approach introduced in Section 6, with the same computational budget of $T = 5 \times 10^9$ and $\tau_1(n_0) = \min(500, \sqrt{n_0})$. Note that the results for ONMC and the reference approach overlap.

We see that the unnested model defines a noticeably different distribution, while the behavior of Anglican is similar, but distinct, to ignoring the **`observe`** statement in the inner query. Further investigation shows that the default behavior of **`conditional`** in a query nesting context is equivalent to using (11) but with N_1 held fixed to at $N_1 = 2$, inducing a substantial bias. More generally, the Anglican source code shows that **`conditional`** defines a Markov chain generated by equalizing the output of the weighted samples generated by running inference on the query. When used to nest queries, this Markov chain is only ever run for a finite length of time, specifically one accept-reject step is carried out, and so does not produce samples from the true conditional distribution.

Plummer (2015) noticed that WinBugs and OpenBugs (Spiegelhalter et al., 1996) similarly do not provide valid inference when using their cut function primitives, which effectively allow the definition of nested inference problems. However, they do not notice the equivalence to the NMC formulation and instead propose a heuristic for reducing the bias that itself has no theoretical guarantees.

4 NESTED CONDITIONING

An alternative way one might wish to nest queries is to use the partition function estimate of one query to factor the trace probability of another. We refer to this as *nested conditioning*. In its simplest form, we can think about conditioning on the values input to the inner query. In Anglican we can carry this out by using the following custom distribution object constructor

```
(defdist nest [inner inputs inf-alg M] []
 (sample [this] nil)
 (observe [this _]
  (log-marginal (take M
   (doquery inf-alg inner inputs)))))
```

When the resulting distribution object is observed, this will now generate, and factor the trace probability by, a partition function estimate for `inner` with inputs `inputs`, constructed using `M` samples of the inference algorithm `inf-alg`. For example, if we were to use the query

```
(defquery outer [D]
 (let [y (sample (beta 2 3))]
  (observe (nest inner [y D] :smc 100) nil)
  y))
```

with `inner` from the nested inference example, then this would form a pseudo marginal sampler (Andrieu and Roberts, 2009) for the unnormalized target distribution

$$\pi_c(y, D) = \textsc{Beta}(y; 2, 3) \int \Gamma(z; y, 1)\, \mathcal{N}(D; y, z^2) dz.$$

Unlike the nested inference case, nested conditioning turns out to be valid even if our budget is held fixed, provided that the partition function estimate is unbiased, as is satisfied by, for example, importance sampling and SMC. In fact, it is important to hold the budget fixed to achieve a MC convergence rate. In general, we can define our target density as

$$p_o(x, y) \propto \pi_o(x, y) = \psi(x, y) p_i(y), \tag{13}$$

where $\psi(x, y)$ is as before (except that we no longer have returned variables from the inner query) and $p_i(y)$ is the true partition function of the inner query when given input y. In practice, we cannot evaluate $p_i(y)$ exactly, but instead produce unbiased estimates $\hat{p}_i(y)$. Using an analogous self-normalized importance sampling to the nested inference case leads to the weights

$$w_n = \psi(x_n, y_n) \hat{p}_i(y_n) / q(x_n, y_n) \tag{14}$$

and corresponding empirical measure

$$\hat{p}(\cdot) = \frac{1}{\sum_{n=1}^{N_0} w_n} \sum_{n=1}^{N_0} w_{n,} \delta_{(x_n, y_n)}(\cdot) \tag{15}$$

such that we are conducting conventional MC estimation, but our weights are now themselves random variables for a given (x_n, y_n) due to the $\hat{p}_i(y_n)$ term. However, the weights are unbiased estimates of the "true weights" $\psi(x_n, y_n) p_i(y_n)/q(x_n, y_n)$ such that we have proper weighting (Naesseth et al., 2015) and thus convergence at the standard MC rate, provided the budget of the inner query remains fixed. This result also follows directly from Theorem 6 of Rainforth et al. (2018), which further ensures no complications arise when conditioning on multiple queries if the corresponding partition function estimates are generated independently. These results further trivially extend to the repeated nesting case by recursion, while using the idea of pseudo-marginal methods (Andrieu and Roberts, 2009), the results also extend to using MCMC based inference for the outermost query.

Rather than just fixing the inputs to the nested query, one can also consider conditioning on the internally sampled variables in the program taking on certain values. Such a nested conditioning approach has been implicitly carried out by Rainforth et al. (2016b); Zinkov and Shan (2017); Scibior and Ghahramani (2016); Ge et al. (2018), each of which manipulate the original program in some fashion to construct a partition function estimator that is used used within a greater inference scheme, e.g. a PMMH estimator (Andrieu et al., 2010).

5 ESTIMATES AS VARIABLES

Our final case is that one might wish to use estimates as first class variables in another query. In other words, a variable in an outer query is assigned to a MC expectation estimate calculated from the outputs of running inference on another, nested, query. By comparison, the nested inference case (without Rao-Blackwellization) can be thought of as assigning a variable in the outer query to a single approximate sample from the conditional distribution of the inner query, rather than an MC expectation estimate constructed by averaging over multiple samples.

Whereas nested inference can only encode a certain class of nested estimation problems – because the only nonlinearity originates from taking the reciprocal of the partition function – using estimates as variables allows, in principle, the encoding of any nested estimation. This is because using the estimate as a first class variable allows arbitrary nonlinear mappings to be applied by the outer query.

An example of this approach is shown in Appendix G, where we construct a generic estimator for Bayesian experimental design problems. Here a partition function estimate is constructed for an inner query and is then used in an outer query. The output of the outer query depends on the logarithm of this estimate, thereby creating the nonlinearity required to form a nested expectation.

Because using estimates as variables allows the encoding of any nested estimation problem, the validity of doing so is equivalent to that of NMC more generally and must thus satisfy the requirements set out in (Rainforth et al., 2018). In particular, one needs to ensure that the budgets used for the inner estimates increase as more samples of the outermost query are taken.

6 ONLINE NESTED MONTE CARLO

NMC will be highly inconvenient to actually implement in a PPS whenever one desires to provide online estimates; for example, a lazy sequence of samples that converges to the target distribution. Suppose that we have already calculated an NMC estimate, but now desire to refine it further. In general, this will require an increase to all N_k for each sample of the outermost estimator. Consequently, the previous samples of the outermost query must be revisited to refine their estimates. This significantly complicates practical implementation, necessitating additional communication between queries, introducing computational overhead, and potentially substantially increasing the memory requirements.

To highlight these shortfalls concretely, consider the nested inference class of problems and, in particular, constructing the un–Rao–Blackwellized estimator (11) in an online fashion. Increasing N_1 requires $m^*(n)$ to be redrawn for each n, which in turn necessitates storage of previous samples and weights.[2] This leads to an overhead cost from the extra computation carried out for revisitation and a memory overhead from having to store information about each call of the inner query.

Perhaps even more problematically, the need to revisit old samples when drawing new samples can cause substantial complications for implementation. Consider implementing such an approach in Anglican. Anglican is designed to return a lazy infinite sequence of samples converging to the target distribution. Once samples are taken from this sequence, they become external to Anglican and cannot be posthumously updated when further samples are requested. Even when all the output samples remain internal, revisiting samples remains difficult: one either needs to implement some form of memory for nested queries so they can be run further, or, if all information is instead stored at the outermost level, additional non-trivial code is necessary to apply post-processing and to revisit queries with previously tested inputs. The latter of these is likely to necessitate inference–algorithm–specific changes, particularly when there are multiple levels of nesting, thereby hampering the entire language construction.

To alleviate these issues, we propose to only increase the computational budget of *new* calls to nested queries, such that earlier calls use fewer samples than later calls. This simple adjustment removes the need for communication between different calls and requires only the storage of the number of times the outermost query has previously been sampled to make updates to the overall estimate. We refer to this approach as *online* NMC (ONMC), which, to the best of our knowledge, has not been previously considered in the literature. As we now show, ONMC only leads to small changes in the convergence rate of the resultant estimator compared to NMC: using recommended parameter settings, the asymptotic root mean squared error for ONMC is never more than twice that of NMC for a matched sample budget and can even be smaller.

Let $\tau_k(n_0) \in \mathbb{N}^+, k = 1, \ldots, D$ be monotonically increasing functions dictating the number of samples used by ONMC at depth k for the n_0-th iteration of the outermost estimator. The ONMC estimator is defined as

$$J_0 = \frac{1}{N_0} \sum_{n_0=1}^{N_0} f_0 \left(y_{n_0}^{(0)}, I_1 \left(y_{n_0}^{(0)}, \tau_{1:D}(n_0) \right) \right) \qquad (16)$$

where $I_1(y_{n_0}^{(0)}, \tau_{1:D}(n_0))$ is calculated using I_1 in (2), setting $y^{(0)} = y_{n_0}^{(0)}$ and $N_k = \tau_k(n_0), \forall k \in 1, \ldots, D$. For reference, the NMC estimator, I_0, is as per (16), except for replacing $\tau_{1:D}(n_0)$ with $\tau_{1:D}(N_0)$. Algorithmically, we have that the ONMC approach is defined as follows.

Algorithm 1 Online Nested Monte Carlo

```
$n_0 \leftarrow 0, \quad J_0 \leftarrow 0$
while true do
    $n_0 \leftarrow n_0 + 1, \quad y_{n_0}^{(0)} \sim p(y^{(0)})$
    Construct $I_1 \left( y_{n_0}^{(0)}, \tau_{1:D}(n_0) \right)$ using $N_k = \tau_k(n_0) \, \forall k$
    $J_0 \leftarrow \frac{n_0 - 1}{n_0} J_0 + f_0 \big( y_{n_0}^{(0)}, I_1 \big( y_{n_0}^{(0)}, \tau_{1:D}(n_0) \big) \big)$
```

We see that OMMC uses fewer samples at inner layers for earlier samples of the outermost level, and that each of resulting inner estimates is calculated as per an NMC estimator with a reduced sample budget. We now show the consistency of the ONMC estimator.

Theorem 2. *If each $\tau_k(n_0) \geq A \left(\log(n_0) \right)^\alpha, \forall n_0 > B$ for some constants $A, B, \alpha > 0$ and each f_k is continuously differentiable, then the mean squared error of J_0 as an estimator for γ_0 converges to zero as $N_0 \to \infty$.*

In other words, ONMC converges for any realistic choice of $\tau_k(n_0)$ provided $\lim_{n_0 \to \infty} \tau_k(n_0) = \infty$: the requirements on $\tau_k(n_0)$ are, for example, much weaker than requiring a logarithmic or faster rate of growth, which would already be an impractically slow rate of increase.

In the case where $\tau_k(n_0)$ increases at a polynomial rate, we can further quantify the rate of convergence, along with the relative variance and bias compared to NMC:

Theorem 3. *If each $\tau_k(n_0) \geq A n_0^\alpha$, $\forall n_0 > B$ for some constants $A, B, \alpha > 0$ and each f_k is continuously differentiable, then*

$$\mathbb{E} \left[(J_0 - \gamma_0)^2 \right] \leq \frac{\varsigma_0^2}{N_0} + \left(\frac{\beta g(\alpha, N_0)}{A N_0^\alpha} \right)^2 + O(\epsilon), \qquad (17)$$

$$\text{where} \quad g(\alpha, N_0) = \begin{cases} 1/(1-\alpha), & \alpha < 1 \\ \log(N_0) + \eta, & \alpha = 1 \\ \zeta(\alpha) N_0^{\alpha - 1}, & \alpha > 1 \end{cases}; \qquad (18)$$

$$\beta = \frac{C_0 \varsigma_1^2}{2} + \sum_{k=0}^{D-2} \left(\prod_{d=0}^{k} K_d \right) \frac{C_{k+1} \varsigma_{k+2}^2}{2}; \qquad (19)$$

$\eta \approx 0.577$ is the Euler–Mascheroni constant; ζ is the

[2] Note that not all previous samples and weights need storing – when making the update we can sample whether to change $m^*(n)$ or not based on combined weights from all the old samples compared to all the new samples.

Riemann–zeta function; and C_k, K_k, and ς_k are constants defined as per the corresponding NMC bound given in (3)*.*

Corollary 1. *Let J_0 be an ONMC estimator setup as per Theorem 3 with N_0 outermost samples and let I_0 be an NMC estimator with a matched overall sample budget. Defining $c = (1+\alpha D)^{(-1/(1+\alpha D))}$, then*

$$\mathrm{Var}[J_0] \to c\mathrm{Var}[I_0] \quad \textit{as} \quad N_0 \to \infty.$$

Further, if the NMC bias decreases at a rate proportional to that implied by the bound given in (3)*, namely*

$$|\mathbb{E}[I_0 - \gamma_0]| = \frac{b}{M_0^\alpha} + O(\epsilon) \tag{20}$$

for some constant $b > 0$, where M_0 is the number of outermost samples used by the NMC sampler, then

$$|\mathbb{E}[J_0 - \gamma_0]| \leq c^\alpha g(\alpha, N_0)\, |\mathbb{E}[I_0 - \gamma_0]| + O(\epsilon).$$

We expect the assumption that the bias scales as $1/M_0^\alpha$ to be satisfied in the vast majority of scenarios, but there may be edge cases, e.g. when an f_k gives a constant output, for which faster rates are observed. Critically, the assumption holds for all nested inference problems because the rate given in (10) is an equality.

We see that if $\alpha < 1$, which will generally be the case in practice for sensible setups, then the convergence rates for ONMC and NMC vary only by a constant factor. Specifically, for a fixed value of N_0, they have the same asymptotic variance and ONMC has a factor of $1/(1-\alpha)$ higher bias. However, the cost of ONMC is (asymptotically) only $c < 1$ times that of NMC, so for a fixed overall sample budget it has lower variance.

As the bound varies only in constant factors for $\alpha < 1$, the asymptotically optimal value for α for ONMC is the same as that for NMC, namely $\alpha = 0.5$ (Rainforth et al., 2018). For this setup, we have $c \in \{0.763, 0.707, 0.693, 0.693, 0.699, 1\}$ respectively for $D \in \{1, 2, 3, 4, 5, \infty\}$. Consequently, when $\alpha = 0.5$, the fixed budget variance of ONMC is always better than NMC, while the bias is no more than 1.75 times larger if $D \leq 13$ and no more than 2 times large more generally.

6.1 EMPIRICAL CONFIRMATION

To test ONMC empirically, we consider the simple analytic model given in Appendix F, setting $\tau_1(n_0) = \max(25, \sqrt{n_o})$. The rationale for setting a minimum value of N_1 is to minimize the burn-in effect of ONMC – earlier samples will have larger bias than later samples and we can mitigate this by ensuring a minimum value for N_1. More generally, we recommend setting (in the absence of other information) $\tau_1(n_0) = \tau_2(n_0) = \cdots = \tau_D(n_0) = \max(T_{\min}^{1/3}, \sqrt{n_0})$, where $T_{\min}$ is the minimum overall budget we expect to spend. In Figure 2, we have chosen to set $T_{\min}$ deliberately low so as to emphasize the differences between NMC and ONMC. Given our value for $T_{\min}$, the ONMC approach is identical to fixing $N_1 = 25$ for $T < 25^3 = 15625$, but unlike fixing N_1, it continues to improve beyond this because it is not limited by asymptotic bias. Instead, we see an inflection point-like behavior around $T_{\min}$, with the rate recovering to effectively match that of the NMC estimator.

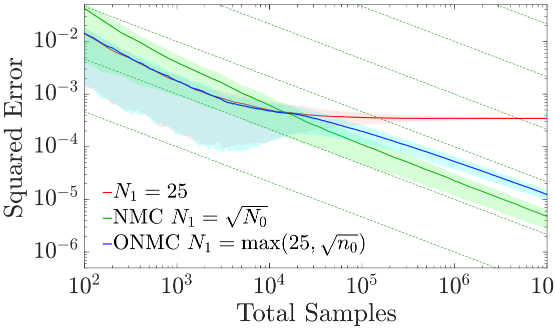

Figure 2: Convergence of ONMC, NMC, and fixed N_1. Results are averaged over 1000 runs, with solid lines showing the mean and shading the 25-75% quantiles. The theoretical rates for NMC are shown by the dashed lines.

6.2 USING ONMC IN PPSs

Using ONMC based estimation schemes to ensure consistent estimation for nested inference in PPSs is straightforward – the number of iterations the outermost query has been run for is stored and used to set the number of iterations used for the inner queries. In fact, even this minimal level of communication is not necessary – n_0 can be inferred from the number of times we have previously run inference on the current query, the current depth k, and $\tau_1(\cdot), \ldots, \tau_{k-1}(\cdot)$.

As with NMC, for nested inference problems ONMC can either return a single sample from each call of a nested query, or Rao–Blackwellize the drawing of this sample when possible. Each respectively produces an estimator analogous to (11) and (9) respectively, except that N_1 in the definition of the inner weights is now a function of n.

Returning to Figure 1, we see that using ONMC with nested importance sampling and only returning a single sample corrects the previous issues with how Anglican deals with nested inference, producing samples indistinguishable from the reference code.

7 CONCLUSIONS

We have formalized the notion of nesting probabilistic program queries and investigated the statistical validity of different categories of nesting. We have found that current systems tend to use methods that lead to asymptotic bias for nested inference problems, but that they are consistent for nested conditioning. We have shown how to carry out the former in a consistent manner and developed a new *online* estimator that simplifies the construction algorithms that satisfy the conditions required for convergence.

References

C. Andrieu and G. O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, pages 697–725, 2009.

C. Andrieu, A. Doucet, and R. Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2010.

G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 1995.

R. Cornish, F. Wood, and H. Yang. Efficient exact inference in discrete Anglican programs. 2017.

K. Csilléry, M. G. Blum, O. E. Gaggiotti, and O. François. Approximate Bayesian Computation (ABC) in practice. *Trends in Ecology & Evolution*, 25(7):410–418, 2010.

M. F. Cusumano-Towner and V. K. Mansinghka. Using probabilistic programs as proposals. *arXiv preprint arXiv:1801.03612*, 2018.

G. Fort, E. Gobet, and E. Moulines. MCMC design-based non-parametric regression for rare-event. application to nested risk computations. *Monte Carlo Methods Appl*, 2017.

H. Ge, K. Xu, and Z. Ghahramani. Turing: a language for composable probabilistic inference. In *AISTATS*, 2018.

N. Goodman, V. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum. Church: a language for generative models. *UAI*, 2008.

N. D. Goodman and A. Stuhlmüller. *The Design and Implementation of Probabilistic Programming Languages*. 2014.

A. D. Gordon, T. A. Henzinger, A. V. Nori, and S. K. Rajamani. Probabilistic programming. In *Proceedings of the on Future of Software Engineering*. ACM, 2014.

R. Hickey. The Clojure programming language. In *Proceedings of the 2008 symposium on Dynamic languages*, page 1. ACM, 2008.

L. J. Hong and S. Juneja. Estimating the mean of a non-linear function of conditional expectation. In *Winter Simulation Conference*, 2009.

T. A. Le, A. G. Baydin, and F. Wood. Nested compiled inference for hierarchical reinforcement learning. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

V. Mansinghka, D. Selsam, and Y. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *arXiv preprint arXiv:1404.0099*, 2014.

T. Mantadelis and G. Janssens. Nesting probabilistic inference. *arXiv preprint arXiv:1112.3785*, 2011.

I. Murray, Z. Ghahramani, and D. J. MacKay. MCMC for doubly-intractable distributions. In *UAI*, 2006.

C. A. Naesseth, F. Lindsten, and T. B. Schön. Nested sequential Monte Carlo methods. In *ICML*, 2015.

L. Ouyang, M. H. Tessler, D. Ly, and N. Goodman. Practical optimal experiment design with probabilistic programs. *arXiv preprint arXiv:1608.05046*, 2016.

M. Plummer. Cuts in Bayesian graphical models. *Statistics and Computing*, 25(1):37–43, 2015.

J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random structures and Algorithms*, 9(1-2): 223–252, 1996.

T. Rainforth. *Automating Inference, Learning, and Design using Probabilistic Programming*. PhD thesis, 2017.

T. Rainforth, R. Cornish, H. Yang, and F. Wood. On the pitfalls of nested Monte Carlo. *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016a.

T. Rainforth, T. A. Le, J.-W. van de Meent, M. A. Osborne, and F. Wood. Bayesian optimization for probabilistic programs. In *NIPS*, pages 280–288, 2016b.

T. Rainforth, R. Cornish, H. Yang, A. Warrington, and F. Wood. On nesting Monte Carlo estimators. In *ICML*, 2018.

A. Scibior and Z. Ghahramani. Modular construction of Bayesian inference algorithms. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

D. Spiegelhalter, A. Thomas, N. Best, and W. Gilks. BUGS 0.5: Bayesian inference using Gibbs sampling manual (version ii). *MRC Biostatistics Unit, Cambridge*, 1996.

A. Stuhlmüller and N. D. Goodman. A dynamic programming algorithm for inference in recursive probabilistic programs. In *Second Statistical Relational AI workshop at UAI 2012 (StaRAI-12)*, 2012.

A. Stuhlmüller and N. D. Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. *Cognitive Systems Research*, 28:80–99, 2014.

D. Tolpin, J.-W. van de Meent, and F. Wood. Probabilistic programming in Anglican. Springer, 2015.

D. Tolpin, J.-W. van de Meent, H. Yang, and F. Wood. Design and implementation of probabilistic programming language Anglican. In *Proceedings of the 28th Symposium on the Implementation and Application of Functional Programming Languages*. ACM, 2016.

F. Wood, J. W. van de Meent, and V. Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, pages 2–46, 2014.

R. Zinkov and C.-C. Shan. Composing inference algorithms as program transformations. In *UAI*, 2017.