astroABC: An Approximate Bayesian Computation Sequential Monte Carlo sampler for cosmological parameter estimation

paper by E. Jennings & M. Madigan

Marco Giunta

21 June 2022

Introduction

From the paper abstract:

"Given the complexity of modern cosmological parameter inference where we are faced with non-Gaussian data and noise, correlated systematics and multi-probe correlated data sets, the Approximate Bayesian Computation (ABC) method is a promising alternative to traditional Markov Chain Monte Carlo approaches in the case where the Likelihood is intractable or unknown. The ABC method is called "Likelihood free" as it avoids explicit evaluation of the Likelihood by using a forward model simulation of the data which can include systematics. We introduce astroABC, an open source ABC Sequential Monte Carlo (SMC) sampler for parameter estimation. [. . .]"

Table of Contents

Table of Contents

Bayesian parameter inference pipeline

Any Bayesian parameter inference problem is made of three ingredients:

$$\text{Prior } \pi(\theta) \rightarrow \text{Likelihood } \mathcal{L}(D|\theta) \rightarrow \text{Posterior } P(\theta|D)$$

Defining each of these terms has its own challenges:

Bayesian parameter inference pipeline

Any Bayesian parameter inference problem is made of three ingredients:

$$\text{Prior } \pi(\theta) \rightarrow \text{Likelihood } \mathcal{L}(D|\theta) \rightarrow \text{Posterior } P(\theta|D)$$

Defining each of these terms has its own challenges:

- How do we effectively incorporate prior knowledge (previous surveys, physical constraints) in the problem, or lack thereof (uninformative priors)?

Bayesian parameter inference pipeline

Any Bayesian parameter inference problem is made of three ingredients:

$$\text{Prior } \pi(\theta) \rightarrow \text{Likelihood } \mathcal{L}(D|\theta) \rightarrow \text{Posterior } P(\theta|D)$$

Defining each of these terms has its own challenges:

- How do we effectively incorporate prior knowledge (previous surveys, physical constraints) in the problem, or lack thereof (uninformative priors)?
- How do we accurately model the problem (theoretical & error contributions)?

Bayesian parameter inference pipeline

Any Bayesian parameter inference problem is made of three ingredients:

$$\text{Prior } \pi(\theta) \rightarrow \text{Likelihood } \mathcal{L}(D|\theta) \rightarrow \text{Posterior } P(\theta|D)$$

Defining each of these terms has its own challenges:

- How do we effectively incorporate prior knowledge (previous surveys, physical constraints) in the problem, or lack thereof (uninformative priors)?
- How do we accurately model the problem (theoretical & error contributions)?
- How do we actually implement Bayes' theorem (MCMC, etc.)?

Bayesian parameter inference pipeline

Any Bayesian parameter inference problem is made of three ingredients:

$$\text{Prior } \pi(\theta) \rightarrow \text{Likelihood } \mathcal{L}(D|\theta) \rightarrow \text{Posterior } P(\theta|D)$$

Defining each of these terms has its own challenges:

- How do we effectively incorporate prior knowledge (previous surveys, physical constraints) in the problem, or lack thereof (uninformative priors)?
- How do we accurately model the problem (theoretical & error contributions)?
- How do we actually implement Bayes' theorem (MCMC, etc.)?

From now on we assume fixing the prior is not an issue & focus on the likelihood

Assembling $\mathcal{L}(\theta|D)$

In order to construct a likelihood appropriate to our problem we must have theoretical results linking variables and a model for the errors involved

Assembling $\mathcal{L}(\theta|D)$

In order to construct a likelihood appropriate to our problem we must have theoretical results linking variables and a model for the errors involved

- Trivial example: 1D linear regression with gaussian errors of fixed variance on y, no errors on x

$$\hat{y}_i = ax_i+b+\varepsilon_i,\ \varepsilon_i \sim N(0,\sigma^2) \implies \mathcal{L}(\vec{y}|a,b) \propto \prod_{i=1}^{N} \exp\left(-\frac{y_i - ax_i - b}{2\sigma^2}\right.$$

Assembling $\mathcal{L}(\theta|D)$

In order to construct a likelihood appropriate to our problem we must have theoretical results linking variables and a model for the errors involved

- Trivial example: 1D linear regression with gaussian errors of fixed variance on y, no errors on x

$$\hat{y}_i = ax_i+b+\varepsilon_i,\ \varepsilon_i \sim N(0,\sigma^2) \implies \mathcal{L}(\vec{y}|a,b) \propto \prod_{i=1}^{N} \exp\left(-\frac{y_i - ax_i - b}{2\sigma^2}\right)$$

- Cosmological example: SN Ia datasets

$$\begin{cases} \mu_i = m_{Bi} - M_i + \alpha x_{1i} + \beta c_i \\ M_i \sim \mathcal{N}(M_0, (\sigma_\mu^{\text{int}})^2) \\ \hat{z}_i \sim \mathcal{N}(z_i, \sigma_z^2) \\ \vdots \end{cases} \implies \mathcal{L}(D|\vec{\theta}) = \dots \ \text{(BHM)}$$

ABC-friendly Likelihoods

Sometimes models have likelihoods that are easy to simulate from, but expensive/impossible to evaluate

ABC-friendly Likelihoods

Sometimes models have likelihoods that are easy to simulate from, but expensive/impossible to evaluate

- Cosmological example: realistic CMB likelihood (anisotropic/correlated noise, sky masking, asymmetric beams $\implies$ mode coupling in Fourier space, no longer diagonal cov. $\implies$ more complicated likelihood), where likelihood approximations are practically mandatory

ABC-friendly Likelihoods

Sometimes models have likelihoods that are easy to simulate from, but expensive/impossible to evaluate

- Cosmological example: realistic CMB likelihood (anisotropic/correlated noise, sky masking, asymmetric beams $\implies$ mode coupling in Fourier space, no longer diagonal cov. $\implies$ more complicated likelihood), where likelihood approximations are practically mandatory
- More general example: non gaussian random field (e.g. we can simulate from a complex stochastic process but computing $p(y|\theta)$ is expensive/impossible as this involves integrating over all possible process realizations; ex. Ising model)

Likelihood-free models

This kind of problem motivated the development of likelihood-free approaches. Intuitive idea:

- If it's easy to simulate from model conditional on parameters...

Likelihood-free models

This kind of problem motivated the development of likelihood-free approaches. Intuitive idea:

- If it's easy to simulate from model conditional on parameters...
- ...we can run simulations for many parameters, then compare the obtained datasets with the observed one.

Likelihood-free models

This kind of problem motivated the development of likelihood-free approaches. Intuitive idea:

- If it's easy to simulate from model conditional on parameters...
- ...we can run simulations for many parameters, then compare the obtained datasets with the observed one.
- The parameter values for which the simulated and observed datasets match are the "good" ones.

Likelihood-free models

This kind of problem motivated the development of likelihood-free approaches. Intuitive idea:

- If it's easy to simulate from model conditional on parameters...
- ...we can run simulations for many parameters, then compare the obtained datasets with the observed one.
- The parameter values for which the simulated and observed datasets match are the "good" ones.

This way we never really explicitly evaluate the likelihood, and yet sample from the true posterior (hopefully). This idea originated in the '80s (*Diggle and Gratton (1984)*, *Rubin (1984, p. 1160)*) and is still used in many modern algorithms, like the one behind *astroABC*

Other uses of Likelihood-free approaches

- As pointed out by *astroABC*'s authors the usefulness of Likelihood-free approaches extends even to situations where likelihood evaluation poses no significant problem.

Other uses of Likelihood-free approaches

- As pointed out by *astroABC*'s authors the usefulness of Likelihood-free approaches extends even to situations where likelihood evaluation poses no significant problem.
- We know that using the wrong likelihood can ruin the quality of the analysis (ex.: biased χ^2 vs rigorous BHM with SN Ia data). What if we're not sure which likelihood is "safe" (ex. artificially skewed normal errors in SN Ia dataset)? As long as we have competing models we can use model selection, but that can be very nontrivial & requires competing models in the first place.

Other uses of Likelihood-free approaches

- As pointed out by *astroABC*'s authors the usefulness of Likelihood-free approaches extends even to situations where likelihood evaluation poses no significant problem.
- We know that using the wrong likelihood can ruin the quality of the analysis (ex.: biased χ^2 vs rigorous BHM with SN Ia data). What if we're not sure which likelihood is "safe" (ex. artificially skewed normal errors in SN Ia dataset)? As long as we have competing models we can use model selection, but that can be very nontrivial & requires competing models in the first place.
- A possible alternative is to skip this issue entirely by simply *not committing to a specific model* via likelihood-free methods. This agnostic approach can be safer, as will be shown later

Table of Contents

Naive Rejection Sampling

The claim that we could sample from the posterior by checking which parameters produce datasets matching to the observed one makes sense intuitively, but we need to show it actually works.

Naive Rejection Sampling

The claim that we could sample from the posterior by checking which parameters produce datasets matching to the observed one makes sense intuitively, but we need to show it actually works.

Naive RS theorem

Say we observed a dataset $\vec{y}$ from which to infer parameters $\vec{\theta}$; our prior is $\pi(\vec{\theta})$ and we can sample datasets $\vec{y}^*$ from the likelihood $\mathcal{L}(\vec{y}|\vec{\theta})$. If we sample $\vec{\theta}^*$ according to the following procedure:

1. $\vec{\theta}^* \sim \pi(\vec{\theta})$
2. $\vec{y}^* \sim \mathcal{L}(\vec{y}|\vec{\theta}^*)$
3. accept $\vec{\theta}^*$ iff $\vec{y}^* = \vec{y}$

then $\vec{\theta}^* \sim P(\vec{\theta}|\vec{y})$, i.e. any $\vec{\theta}^*$ obtained as above is sampled from the true posterior.

Proof of NRS theorem I

To prove the theorem we simply need to compute the probability of accepting a set of parameters; if we can show this equals the posterior then we're done.

Proof of NRS theorem I

To prove the theorem we simply need to compute the probability of accepting a set of parameters; if we can show this equals the posterior then we're done.
Clearly the probability of sampling $\vec{\theta}^*$ can be computed as the marginal of its joint with any $\vec{y}^*$:

$$P(\vec{\theta}^*) = \int \mathrm{d}\vec{y}^* \, P(\vec{\theta}^*, \vec{y}^*)$$

We marginalize because we want the probability of sampling $\vec{\theta}^*$ irrespective of its "partner dataset", which of course in principle are infinite in number.

Proof of NRS theorem II

In order for a specific pair $(\vec{\theta}^*, \vec{y}^*)$ to be accepted according to the above algorithm three things must happen:

Proof of NRS theorem II

In order for a specific pair $(\vec{\theta}^*, \vec{y}^*)$ to be accepted according to the above algorithm three things must happen:

1. $\vec{\theta}^*$ must be sampled from the prior $\pi(\vec{\theta}) \implies$ prob. $= \pi(\vec{\theta}^*)$;

Proof of NRS theorem II

In order for a specific pair $(\vec{\theta}^*, \vec{y}^*)$ to be accepted according to the above algorithm three things must happen:

1. $\vec{\theta}^*$ must be sampled from the prior $\pi(\vec{\theta}) \implies$ prob. $= \pi(\vec{\theta}^*)$;
2. once $\vec{\theta}^*$ has been sampled $\vec{y}^*$ must be sampled from the likelihood conditioned on $\vec{\theta}^*$, i.e. from $\mathcal{L}(\vec{y}|\vec{\theta}^*) \implies$ prob. $= \mathcal{L}(\vec{y}^*|\vec{\theta}^*)$;

Proof of NRS theorem II

In order for a specific pair $(\vec{\theta}^*, \vec{y}^*)$ to be accepted according to the above algorithm three things must happen:

1. $\vec{\theta}^*$ must be sampled from the prior $\pi(\vec{\theta}) \implies$ prob. $= \pi(\vec{\theta}^*)$;
2. once $\vec{\theta}^*$ has been sampled $\vec{y}^*$ must be sampled from the likelihood conditioned on $\vec{\theta}^*$, i.e. from $\mathcal{L}(\vec{y}|\vec{\theta}^*) \implies$ prob. $= \mathcal{L}(\vec{y}^*|\vec{\theta}^*)$;
3. finally it must be $\vec{y}^* = \vec{y} \implies$ prob. $= \delta(\vec{y}^* - \vec{y})$ (due to $\vec{y}^* \neq \vec{y} \implies P(\vec{\theta}^*, \vec{y}^*) = 0$ automatically + normalization along $\vec{y}$ axis).

Proof of NRS theorem II

In order for a specific pair $(\vec{\theta}^*, \vec{y}^*)$ to be accepted according to the above algorithm three things must happen:

1. $\vec{\theta}^*$ must be sampled from the prior $\pi(\vec{\theta}) \implies$ prob. $= \pi(\vec{\theta}^*)$;
2. once $\vec{\theta}^*$ has been sampled $\vec{y}^*$ must be sampled from the likelihood conditioned on $\vec{\theta}^*$, i.e. from $\mathcal{L}(\vec{y}|\vec{\theta}^*) \implies$ prob. $= \mathcal{L}(\vec{y}^*|\vec{\theta}^*)$;
3. finally it must be $\vec{y}^* = \vec{y} \implies$ prob. $= \delta(\vec{y}^* - \vec{y})$ (due to $\vec{y}^* \neq \vec{y} \implies P(\vec{\theta}^*, \vec{y}^*) = 0$ automatically + normalization along $\vec{y}$ axis).

These three events must all happen, therefore the final probability its their product. By exploiting the above we therefore obtain:

$$P(\vec{\theta}^*, \vec{y}^*) = \underbrace{\pi(\vec{\theta}^*)}_{1}\underbrace{\mathcal{L}(\vec{y}^*|\vec{\theta}^*)}_{2}\underbrace{\delta(\vec{y}^* - \vec{y})}_{3}$$

Proof of NRS theorem III

Finally by integrating, then using Bayes' theorem we trivially obtain the result.

$$P(\vec{\theta}^*) = \int \mathrm{d}\vec{y}^* \, \mathcal{L}(\vec{y}^*|\vec{\theta}^*)\pi(\vec{\theta}^*)\delta(\vec{y}^* - \vec{y}) = \mathcal{L}(\vec{y}|\vec{\theta}^*)\pi(\vec{\theta}^*) \propto P(\vec{\theta}^*|\vec{y})$$

hence all accepted $\vec{\theta}^*$ are drawn from the exact posterior.
This can also be proved in the reverse order starting from the posterior, then adding the Dirac delta and interpreting each term inside the integral.
(Proof adapted from [2] & personal notes)

From Naive Rejection sampling to Rejection Sampling

- The above result is exact and can be turned into an embarassingly parallel algorithm (impossible e.g. with MH/Gibbs), but can never work in practice in most situations - unless we're dealing with discrete distributions over limited domains, that is. If our parameters are real numbers it's almost impossible to randomly achieve $\vec{y}^* = \vec{y}$ exactly (especially when dealing with floating representation of real numbers in software)

From Naive Rejection sampling to Rejection Sampling

- The above result is exact and can be turned into an embarassingly parallel algorithm (impossible e.g. with MH/Gibbs), but can never work in practice in most situations - unless we're dealing with discrete distributions over limited domains, that is. If our parameters are real numbers it's almost impossible to randomly achieve $\vec{y}^* = \vec{y}$ exactly (especially when dealing with floating representation of real numbers in software)
- This means our Bayesian computation must become approximate (hence ABC) in the sense that we replace the exact condition $\vec{y}^* = \vec{y}$ with an approximate one: $\vec{y}^* \approx \vec{y}$

From Naive Rejection sampling to Rejection Sampling

- The above result is exact and can be turned into an embarassingly parallel algorithm (impossible e.g. with MH/Gibbs), but can never work in practice in most situations - unless we're dealing with discrete distributions over limited domains, that is. If our parameters are real numbers it's almost impossible to randomly achieve $\vec{y}^* = \vec{y}$ exactly (especially when dealing with floating representation of real numbers in software)
- This means our Bayesian computation must become approximate (hence ABC) in the sense that we replace the exact condition $\vec{y}^* = \vec{y}$ with an approximate one: $\vec{y}^* \approx \vec{y}$
- To achieve this we must define an appropriate metric ρ in our data space (e.g. weighted L2/L1 norm is a common choice, see [1]) and a tolerance parameter $\varepsilon > 0$, s.t. $\vec{y}^* \approx \vec{y} \iff \rho(\vec{y}^*, \vec{y}) < \varepsilon$

Approximate Rejection Sampling

It's trivial to prove that the approximate procedure works as expected.

Approximate RS theorem

Say we observed a dataset $\vec{y}$ from which to infer parameters $\vec{\theta}$; our prior is $\pi(\vec{\theta})$ and we can sample datasets $\vec{y}^*$ from the likelihood $\mathcal{L}(\vec{y}|\vec{\theta})$. If we sample $\vec{\theta}^*$ according to the following procedure:

1. $\vec{\theta}^* \sim \pi(\vec{\theta})$
2. $\vec{y}^* \sim \mathcal{L}(\vec{y}|\vec{\theta}^*)$
3. accept $\vec{\theta}^*$ iff $\rho(\vec{y}^* - \vec{y}) < \varepsilon$ for some small $\varepsilon > 0$

then $\vec{\theta}^* \sim P_\varepsilon(\vec{\theta}|\vec{y})$ s. t. $\lim_{\varepsilon \to 0} P_\varepsilon(\vec{\theta}|\vec{y}) = P(\vec{\theta}|\vec{y})$, i.e. any $\vec{\theta}^*$ obtained as above is sampled from *an approximation of* the true posterior.

Proof of Approximate RS

The proof (adapted from [2]) is identical; the only change is that the last probability now is over a (small) interval of values, therefore we need to replace the Dirac delta with the indicator/characteristic function of set $A_{\vec{y},\varepsilon} = \{\vec{y}^* | \rho(\vec{y}^*, \vec{y}) < \varepsilon\}$.

$$P(\vec{\theta}^*) = \int d\vec{y}^* \, \mathcal{L}(\vec{y}^* | \vec{\theta}^*) \pi(\vec{\theta}^*) \mathbb{1}(\rho(\vec{y}^*, \vec{y}) < \varepsilon) \equiv P_\varepsilon(\vec{\theta}^* | \vec{y})$$

Intuitively if ε is small enough then the interval is so small that all $\vec{y}^*$ inside are approximately equal to $\vec{y} =$ const., which makes the integral approximately equal to $\mathcal{L}(\vec{y} | \vec{\theta}^*) \pi(\vec{\theta}^*) \propto P(\vec{\theta}^* | \vec{y})$; more formally

$$\lim_{\varepsilon \to 0} \mathbb{1}(\rho(\vec{y}^*, \vec{y}) < \varepsilon) = \delta(\vec{y}^* - \vec{y})$$

from which we trivially obtain the final result:

$$\lim_{\varepsilon \to 0} P_\varepsilon(\vec{\theta}^* | \vec{y}) = P(\vec{\theta}^* | \vec{y}) \iff P_\varepsilon(\vec{\theta}^* | \vec{y}) = P(\vec{\theta}^* | \rho(\vec{y}^* - \vec{y}) < \varepsilon)$$

ε parameter interpretation

Notice that if $\varepsilon \to +\infty$ then every $\vec{\theta}^*$ sampled from the prior is accepted; therefore in this limit we're simply reconstructing the prior by exactly sampling from it, i.e. we're learning nothing. On the other hand for $\varepsilon = 0$ we sample from the posterior exactly and for $\varepsilon \approx 0$ we sample from an approximation of the posterior. Adapting [2]:

ε parameter interpretation

Notice that if $\varepsilon \to +\infty$ then every $\vec{\theta}^*$ sampled from the prior is accepted; therefore in this limit we're simply reconstructing the prior by exactly sampling from it, i.e. we're learning nothing. On the other hand for $\varepsilon = 0$ we sample from the posterior exactly and for $\varepsilon \approx 0$ we sample from an approximation of the posterior. Adapting [2]:

Important ABC result

Convergence in distribution:

1. $\varepsilon \to 0 \implies P_\varepsilon(\vec{\theta}|\vec{y}) \to P(\vec{\theta}|\vec{y})$
2. $\varepsilon \to +\infty \implies P_\varepsilon(\vec{\theta}|\vec{y}) \to \pi(\theta)$

i.e. when ε is too large we learn nothing.

ε parameter interpretation

Notice that if $\varepsilon \to +\infty$ then every $\vec{\theta}^*$ sampled from the prior is accepted; therefore in this limit we're simply reconstructing the prior by exactly sampling from it, i.e. we're learning nothing. On the other hand for $\varepsilon = 0$ we sample from the posterior exactly and for $\varepsilon \approx 0$ we sample from an approximation of the posterior. Adapting [2]:

Important ABC result

Convergence in distribution:

1. $\varepsilon \to 0 \implies P_\varepsilon(\vec{\theta}|\vec{y}) \to P(\vec{\theta}|\vec{y})$
2. $\varepsilon \to +\infty \implies P_\varepsilon(\vec{\theta}|\vec{y}) \to \pi(\theta)$

i.e. when ε is too large we learn nothing.

And yet if ε is too small too many samples are rejected and the algorithm becomes inefficient, so some balance must be found (e.g. by annealing ε).

RS “leftover” inefficiencies

With an appropriate ε approximation the RS algorithm can now work in any practical situation, and yet it turns out it’s very inefficient i.e. a huge number of candidates are rejected. As discussed in the previous slide this is in part unavoidable (otherwise no learning can happen, in contrast e.g. to MCMC), and yet there are other factors that make the acceptance rate even worse than it needs to be:

RS “leftover” inefficiencies

With an appropriate ε approximation the RS algorithm can now work in any practical situation, and yet it turns out it's very inefficient i.e. a huge number of candidates are rejected. As discussed in the previous slide this is in part unavoidable (otherwise no learning can happen, in contrast e.g. to MCMC), and yet there are other factors that make the acceptance rate even worse than it needs to be:

1. *Problem:* due to the curse of dimensionality it's unlikely that two random points happen to be close; *possible solution:* summary statistics.

RS “leftover” inefficiencies

With an appropriate ε approximation the RS algorithm can now work in any practical situation, and yet it turns out it’s very inefficient i.e. a huge number of candidates are rejected. As discussed in the previous slide this is in part unavoidable (otherwise no learning can happen, in contrast e.g. to MCMC), and yet there are other factors that make the acceptance rate even worse than it needs to be:

1. *Problem:* due to the curse of dimensionality it’s unlikely that two random points happen to be close; *possible solution:* summary statistics.
2. *Problem:* Unless the prior is already close to the posterior many sampled points will be in the “wrong” region; *possible solution:* SMC.

The need for Summary Statistics I

We know that the volume of e.g. the unit hypersphere scales exponentially with space dimensionality; this is due to the *curse of dimensionality*: in higher dimensional spaces there's more "space" available. An important consequence of this is that on average randomly sampled points are more distant, and in particular the fraction of points randomly sampled within a small region (e.g. ε distance away from $\vec{y}$) becomes exponentially smaller at high d. This is the same issue that makes pure MC methods impractical with large d.

The need for Summary Statistics II

A way to fight against this curse is to first project sampled points onto a smaller dimensional space, where there's intrinsically less available "space"; this way more points will happen to be closer to our target, hence improving the acceptance rate.
This projection of the data onto a lower dimensional space is called *summary statistics*, and in general is a function $S : D \to D_*$ which maps our dataset D into a compressed version D_*.

Sufficient Summary Statistics I

Of course in order not to lose any information this projection must be invertible; in this way the compression is *lossless*, i.e. from the compressed dataset we can always recover the same information about the parameters of interest contained in the original data.
This leads us to *sufficient summary statistics*, which are function which satisfy this property. More formally:

Sufficient Summary Statistics I

Of course in order not to lose any information this projection must be invertible; in this way the compression is *lossless*, i.e. from the compressed dataset we can always recover the same information about the parameters of interest contained in the original data.
This leads us to *sufficient summary statistics*, which are function which satisfy this property. More formally:

Sufficient Summary Statistics

A summary statistics S is sufficient for $\vec{\theta}$ if

$$P_\varepsilon(\vec{\theta}|S(\vec{y})) = P_\varepsilon(\vec{\theta}|\vec{y})$$

i.e. any information about $\vec{\theta}$ contained in $\vec{y}$ is also contained in $S(\vec{y})$, in the sense that we can infer the same posterior about $\vec{\theta}$ whether we use the original or the compressed dataset.

Sufficient Summary Statistics II

An example of summary statistics is the sum and sum of squares of points sampled from a 1D normal distribution. We know a distribution of this kind is completely specified by its mean and variance, whose frequentist ML estimators are assembled using $\sum_i x_i,\ \sum_i x_i^2$; similarly in a Bayesian setting if we use conjugate priors the posteriors of these parameters are fully specified by these two quantities (see [4]).

Sufficient Summary Statistics III

Unfortunately the only guaranteed/easy to find sufficient summary statistics are for members of the exponential family (e.g. normal distributions); for this reason one usually looks for "almost lossless" compressions, e.g. those that minimize the *reconstruction error*

$$\rho_S(S^{-1}(S(\vec{y})) - \vec{y})$$

according to some distance measure ρ_S for invertible functions S.

Sufficient Summary Statistics III

Unfortunately the only guaranteed/easy to find sufficient summary statistics are for members of the exponential family (e.g. normal distributions); for this reason one usually looks for “almost lossless” compressions, e.g. those that minimize the *reconstruction error*

$$\rho_S(S^{-1}(S(\vec{y})) - \vec{y})$$

according to some distance measure ρ_S for invertible functions S.
In general the problem of constructing “informative enough” statistics as a replacement for unattainable sufficient ones is a central topic and an open problem in ABC. If we find a good enough summary we can bypass the curse of dimensionality problem, and make ABC more efficient.

Sufficient Summary Statistics IV

We can now update the RS algorithm by introducing summary statistics. The procedure to sample is now the following:

Sufficient Summary Statistics IV

We can now update the RS algorithm by introducing summary statistics. The procedure to sample is now the following:

Approximated RS with summary statistics

1. $\vec{\theta}^* \sim \pi(\vec{\theta}^*)$
2. $\vec{y}^* \sim \mathcal{L}(\vec{y}|\vec{\theta}^*)$
3. accept $\vec{\theta}^*$ iff $\rho(S(\vec{y}^*) - S(\vec{y})) < \varepsilon$

As long as S is a sufficient summary statistics $\vec{\theta}^*$ will be sampled from the same (approximation of the) posterior by definition of sufficient statistics; otherwise we add an extra layer of approximation.

Sufficient Summary Statistics IV

We can now update the RS algorithm by introducing summary statistics. The procedure to sample is now the following:

Approximated RS with summary statistics

1. $\vec{\theta}^* \sim \pi(\vec{\theta}^*)$
2. $\vec{y}^* \sim \mathcal{L}(\vec{y}|\vec{\theta}^*)$
3. accept $\vec{\theta}^*$ iff $\rho(S(\vec{y}^*) - S(\vec{y})) < \varepsilon$

As long as S is a sufficient summary statistics $\vec{\theta}^*$ will be sampled from the same (approximation of the) posterior by definition of sufficient statistics; otherwise we add an extra layer of approximation.

For this reason ABC is in practice an "approximate bayesian computation" not just in the sense of ε but also of S.

Beyond simple Rejection Sampling I

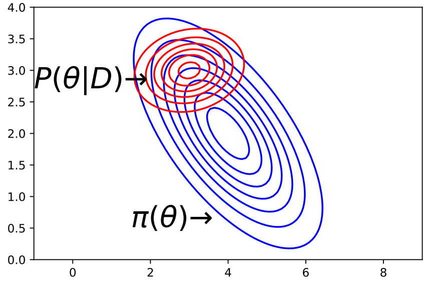
There is one final source of inefficiency tackled by the *astroABC* main algorithm, and it can be explained by adapting from [2]: *[...] Parameters are proposed from prior $\pi(\vec{\theta})$, which does not exploit the information of already accepted parameters. Unless $\pi(\vec{\theta})$ is somehow [already] similar to $P_\varepsilon(\vec{\theta}|\vec{y})$ many proposals will be rejected for moderately small ε [...]. [Therefore] a natural approach is to consider ABC within an MCMC algorithm, [...so that] the proposed parameter explores a neighbourhood of the last accepted parameter.*

Beyond simple Rejection Sampling II

Usually the prior and the posterior are quite different (otherwise inference is useless), so when sampling from the prior there's a lot of "wasted space" from which parameters are often discarded. Let us visualize this:

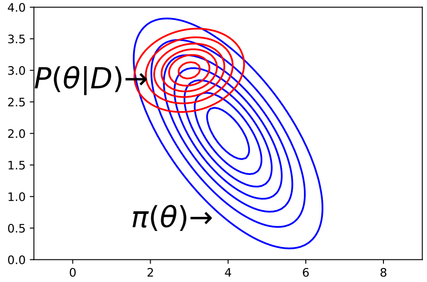
Beyond simple Rejection Sampling II

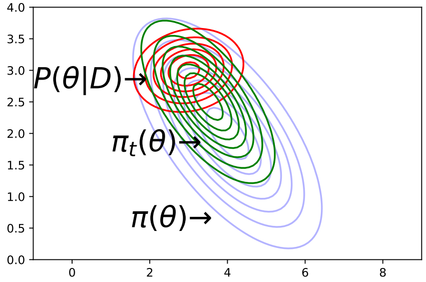
Usually the prior and the posterior are quite different (otherwise inference is useless), so when sampling from the prior there's a lot of "wasted space" from which parameters are often discarded. Let us visualize this:



Beyond simple Rejection Sampling II

Usually the prior and the posterior are quite different (otherwise inference is useless), so when sampling from the prior there's a lot of "wasted space" from which parameters are often discarded. Let us visualize this:





To improve the efficiency we can use what we learned up to time t to build a perturbed prior $\pi_t(\vec{\theta})$, which more closely resembles the final posterior

Particle-based ABC algorithms

- As we noted before ABC methods lend themselves to be parallelized, since each proposal can be sampled and evaluated independently of others being obtained at the same time.

Particle-based ABC algorithms

- As we noted before ABC methods lend themselves to be parallelized, since each proposal can be sampled and evaluated independently of others being obtained at the same time.
- An efficient parallelization scheme (which is also "intermediate distributions friendly") is *particle-based sampling*.

Particle-based ABC algorithms

- As we noted before ABC methods lend themselves to be parallelized, since each proposal can be sampled and evaluated independently of others being obtained at the same time.
- An efficient parallelization scheme (which is also "intermediate distributions friendly") is *particle-based sampling*.
- We call each sampled parameter a "particle". The idea is that we start with a population of particles, then we let them evolve through time so that at each iteration their distribution is closer and closer to the final posterior approximant.

Particle-based ABC algorithms

- As we noted before ABC methods lend themselves to be parallelized, since each proposal can be sampled and evaluated independently of others being obtained at the same time.
- An efficient parallelization scheme (which is also "intermediate distributions friendly") is *particle-based sampling*.
- We call each sampled parameter a "particle". The idea is that we start with a population of particles, then we let them evolve through time so that at each iteration their distribution is closer and closer to the final posterior approximant.
- This means that from the particle population at time t we can construct the intermediate distribution π_t, improve it some way, then resample the population of particles from the new distribution, which is closer to the target.

Sequential Importance Sampling (SIS) I

A possible way to implement the above solution is to build a sequence of distribution $\{\pi_t(\theta)\},\ t = 1, \dots, T$ using *Sequential Importance Sampling* (SIS). This algorithm works as follows (adapted from [3]):

- Let's say we start from a distribution $\pi_1(\theta)$ (e.g. the prior) and we want to get closer and closer to the final distribution π_T (e.g. $P_\varepsilon(\theta|y)$, the posterior approximant) in T steps.

Sequential Importance Sampling (SIS) I

A possible way to implement the above solution is to build a sequence of distribution $\{\pi_t(\theta)\},\ t = 1, \ldots, T$ using *Sequential Importance Sampling* (SIS). This algorithm works as follows (adapted from [3]):

- Let's say we start from a distribution $\pi_1(\theta)$ (e.g. the prior) and we want to get closer and closer to the final distribution π_T (e.g. $P_\varepsilon(\theta|y)$, the posterior approximant) in T steps.
- In order to get closer and closer to π_T the π_t distribution should improve the information contained in π_{t-1}; in the context of SIS/SMC this intuitive idea can be translated e.g. into minimizing the Kullback-Leiber divergence with the target (see e.g. [1]).

Sequential Importance Sampling (SIS) I

A possible way to implement the above solution is to build a sequence of distribution $\{\pi_t(\theta)\},\ t = 1, \dots, T$ using *Sequential Importance Sampling* (SIS). This algorithm works as follows (adapted from [3]):

- Let's say we start from a distribution $\pi_1(\theta)$ (e.g. the prior) and we want to get closer and closer to the final distribution π_T (e.g. $P_\varepsilon(\theta|y)$, the posterior approximant) in T steps.
- In order to get closer and closer to π_T the π_t distribution should improve the information contained in π_{t-1}; in the context of SIS/SMC this intuitive idea can be translated e.g. into minimizing the Kullback-Leiber divergence with the target (see e.g. [1]).
- Since we're dealing with an evolving population of interactive particles we need to *sample* from each current distribution; this can be achieved by defining π_t for any t with *importance sampling*.

Sequential Importance Sampling (SIS) II

The idea behind importance sampling is that instead of sampling from π_t directly we sample from η_t, which ideally is a distribution close to π_t but easier to sample from; then in order to "bend" η_t and turn it into π_t we rescale probabilities using *importance weights*.
Let us see how this is implemented in practice.

Sequential Importance Sampling (SIS) III

- At time t we sample proposals from a population $\{\theta_t^{(i)}\}_{i=1}^N$ of the $i = 1, \ldots, N$ particles from π_{t-1}, then perturb them according to η_t until we find accepted proposals;

Sequential Importance Sampling (SIS) III

- At time t we sample proposals from a population $\{\theta_t^{(i)}\}_{i=1}^N$ of the $i = 1, \ldots, N$ particles from π_{t-1}, then perturb them according to η_t until we find accepted proposals;
- then we evaluate the weights $w_t^{(i)}(\theta_t^{(i)}) = \pi_t(\theta_t^{(i)})/\eta_t(\theta_t^{(i)})$ with the accepted proposals i.e. the updated population.

Sequential Importance Sampling (SIS) III

- At time t we sample proposals from a population $\{\theta_t^{(i)}\}_{i=1}^N$ of the $i = 1, \dots, N$ particles from π_{t-1}, then perturb them according to η_t until we find accepted proposals;
- then we evaluate the weights $w_t^{(i)}(\theta_t^{(i)}) = \pi_t(\theta_t^{(i)})/\eta_t(\theta_t^{(i)})$ with the accepted proposals i.e. the updated population.
- These weights are the discretized version of π_t, which can be used to sample the to-be-perturbed particle population at the beginning of the next iteration.

Sequential Importance Sampling (SIS) III

- At time t we sample proposals from a population $\{\theta_t^{(i)}\}_{i=1}^N$ of the $i = 1, \ldots, N$ particles from π_{t-1}, then perturb them according to η_t until we find accepted proposals;
- then we evaluate the weights $w_t^{(i)}(\theta_t^{(i)}) = \pi_t(\theta_t^{(i)})/\eta_t(\theta_t^{(i)})$ with the accepted proposals i.e. the updated population.
- These weights are the discretized version of π_t, which can be used to sample the to-be-perturbed particle population at the beginning of the next iteration.

SIS is a general framework where one is free to choose the distribution sequences $\{\pi_t\}_{t=1}^T$ and $\{\eta_t\}_{t=1}^T$. *ABC Sequential Monte Carlo* (ABC SMC) is the variant where a) π_1 is the prior, the target π_T is the posterior approximant, and each in-between distribution is a more and more "carved out" prior, and b) each approximant η_t is a *Monte Carlo* estimate of a perturbed version of the previous distribution π_{t-1}.

ABC Sequential Monte Carlo (ABC SMC) I

π_t *sequence:* In ABC SMC we define:

$$\begin{cases} \pi_1(\theta) = \pi(\theta) = \text{prior} \\ \pi_t(\theta) = \pi(\theta)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \; t = 2, \ldots, T-1 \\ \pi_T(\theta) = P_\varepsilon(\theta|y) = \text{posterior approximant} \end{cases}$$

In the above $y(\theta)$ is a shorthand notation to mean that y is sampled from the likelihood conditioned on θ, i.e. $y(\theta) \sim \mathcal{L}(y|\theta)$.

ABC Sequential Monte Carlo (ABC SMC) I

π_t *sequence:* In ABC SMC we define:

$$\begin{cases} \pi_1(\theta) = \pi(\theta) = \text{prior} \\ \pi_t(\theta) = \pi(\theta)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \ t = 2, \dots, T-1 \\ \pi_T(\theta) = P_\varepsilon(\theta|y) = \text{posterior approximant} \end{cases}$$

In the above $y(\theta)$ is a shorthand notation to mean that y is sampled from the likelihood conditioned on θ, i.e. $y(\theta) \sim \mathcal{L}(y|\theta)$.
Interpretation: π_t is a "carved out" prior. If a specific $\tilde{\theta}$ is incapable of generating a dataset matching the observed one its new probability gets set equal to zero, even if its prior probability is $\pi(\tilde{\theta}) \neq 0$. This way we progressively "remove" sections of the prior and get closer to the "smaller" posterior distribution.

ABC Sequential Monte Carlo (ABC SMC) II

η_t *sequence:* In ABC SMC we define:

$$\begin{cases} \eta_1(\theta) = \pi_1(\theta) = \pi(\theta) \\ \eta_t(\theta_t) = \mathbb{1}(\pi(\theta_t) > 0)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \int \pi_{t-1}(\theta_{t-1}) K_t(\theta_{t-1}, \theta_t)\, \mathrm{d}\theta_{t-1} \end{cases}$$

where we exploited that to get the importance weights we only need to define/compute η_t in θ_t, not in an arbitrary θ.

ABC Sequential Monte Carlo (ABC SMC) II

η_t *sequence:* In ABC SMC we define:

$$\begin{cases} \eta_1(\theta) = \pi_1(\theta) = \pi(\theta) \\ \eta_t(\theta_t) = \mathbb{1}(\pi(\theta_t) > 0)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \int \pi_{t-1}(\theta_{t-1}) K_t(\theta_{t-1}, \theta_t)\, \mathrm{d}\theta_{t-1} \end{cases}$$

where we exploited that to get the importance weights we only need to define/compute η_t in θ_t, not in an arbitrary θ.
Interpretation: $\eta_t(\theta_t)$ is automatically set to 0 if θ_t has 0 prior probability and/or is incapable of producing a matching dataset; otherwise it's the perturbed version of the previous distribution π_{t-1} via a perturbation kernel K_t.

ABC Sequential Monte Carlo (ABC SMC) II

η_t *sequence:* In ABC SMC we define:

$$\begin{cases} \eta_1(\theta) = \pi_1(\theta) = \pi(\theta) \\ \eta_t(\theta_t) = \mathbb{1}(\pi(\theta_t) > 0)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \int \pi_{t-1}(\theta_{t-1}) K_t(\theta_{t-1}, \theta_t) \, \mathrm{d}\theta_{t-1} \end{cases}$$

where we exploited that to get the importance weights we only need to define/compute η_t in θ_t, not in an arbitrary θ.
Interpretation: $\eta_t(\theta_t)$ is automatically set to 0 if θ_t has 0 prior probability and/or is incapable of producing a matching dataset; otherwise it's the perturbed version of the previous distribution π_{t-1} via a perturbation kernel K_t.
$\eta_t(\theta_t)$ *evaluation:* notice that
$\int \pi_{t-1}(\theta_{t-1}) K_t(\theta_{t-1}, \theta_t) \, \mathrm{d}\theta_{t-1} = \langle K_t(\theta_{t-1}, \theta_t) \rangle_{\pi_{t-1}}$, which can be approximated as a Monte Carlo average (*Sequential Monte Carlo*).

ABC Sequential Monte Carlo (ABC SMC) III

Monte Carlo computation of $\eta_t(\theta_t)$:

$$\begin{aligned}
\eta_t(\theta_t) &= \mathbb{1}(\pi(\theta_t) > 0)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t)\,\langle K_t(\theta_{t-1}, \theta_t)\rangle_{\pi_{t-1}} \\
&\approx \mathbb{1}(\pi(\theta_t) > 0)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \cdot \frac{1}{N} \sum_{\theta_{t-1}^{(j)} \sim \pi_{t-1}} K_t(\theta_{t-1}^{(j)}, \theta_t) \\
&= \mathbb{1}(\pi(\theta_t) > 0)\mathbb{1}(\rho(y(\theta), y_{\text{obs}}) < \varepsilon_t) \cdot \frac{1}{N} \sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t)
\end{aligned}$$

where N is the n. of particles, and $\{\theta_{t-1}^{(j)}\}_{j=1}^{N}$ is the population of particles from intermediate distribution π_{t-1}.

ABC Sequential Monte Carlo (ABC SMC) IV

Finally we can write the unnormalized weights for all *accepted* particles θ_t at any time $t > 1$:

$$w_t(\theta_t) = \frac{\pi(\theta_t)}{\eta_t(\theta_t)} = \frac{\pi(\theta_t)}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t)}$$

We now have all elements to write down the final ABC SMC algorithm employed in the `astroABC` package; adapting again from [3] we write down the final algorithm.

ABC SMC algorithm I

1. Initialize $\varepsilon_1, \ldots, \varepsilon_t$.

ABC SMC algorithm I

1. Initialize $\varepsilon_1, \dots, \varepsilon_t$.
2. For each particle sample a proposal θ^{**} according to the following scheme:

ABC SMC algorithm I

1. Initialize $\varepsilon_1, \ldots, \varepsilon_t$.
2. For each particle sample a proposal θ^{**} according to the following scheme:
    - If $t = 1$ $\theta^{**} \sim \pi(\theta)$;

ABC SMC algorithm I

1. Initialize $\varepsilon_1, \dots, \varepsilon_t$.
2. For each particle sample a proposal θ^{**} according to the following scheme:
   - If $t = 1$ $\theta^{**} \sim \pi(\theta)$;
   - If $t > 1$ first sample θ^* from the previous population $\{\theta_{t-1}^{(i)}\}$ *with probabilities given by the weights* w_{t-1} and perturb the resulting particle to obtain $\theta^{**} \sim K_t(\theta|\theta^*)$, where K_t is a perturbation kernel.

ABC SMC algorithm I

1. Initialize $\varepsilon_1, \dots, \varepsilon_t$.
2. For each particle sample a proposal θ^{**} according to the following scheme:
    - If $t = 1$ $\theta^{**} \sim \pi(\theta)$;
    - If $t > 1$ first sample θ^{*} from the previous population $\{\theta^{(i)}_{t-1}\}$ *with probabilities given by the weights* w_{t-1} and perturb the resulting particle to obtain $\theta^{**} \sim K_t(\theta|\theta^{*})$, where K_t is a perturbation kernel.

   If $\pi(\theta^{**}) = 0$ resample θ^{**}; otherwise simulate a candidate dataset $y^{**} \sim \mathcal{L}(y|\theta^{**})$. If the condition $\rho(y^{**} - y_{\text{obs}}) < \varepsilon_t$ isn't met resample θ^{**}.

ABC SMC algorithm II

3. For each particle i set $\theta_t^{(i)}$ equal to their proposal $\theta^{**(i)}$ sampled in the previous step, then for (accepted) particle $\theta_t^{(i)}$ calculate the weight

$$w_t^{(i)} = \begin{cases} 1 & \text{if } t = 1 \\ \frac{\pi(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})} & \text{if } t > 1 \end{cases}$$

ABC SMC algorithm II

3. For each particle i set $\theta_t^{(i)}$ equal to their proposal $\theta^{**(i)}$ sampled in the previous step, then for (accepted) particle $\theta_t^{(i)}$ calculate the weight

$$w_t^{(i)} = \begin{cases} 1 & \text{if } t = 1 \\ \frac{\pi(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})} & \text{if } t > 1 \end{cases}$$

4. Once all weights have been computed normalize them to 1, then advance to next iteration $t + 1$.

ABC SMC algorithm II

3. For each particle i set $\theta_t^{(i)}$ equal to their proposal $\theta^{**(i)}$ sampled in the previous step, then for (accepted) particle $\theta_t^{(i)}$ calculate the weight

$$w_t^{(i)} = \begin{cases} 1 & \text{if } t = 1 \\ \frac{\pi(\theta_t^{(i)})}{\sum_{j=1}^{N} w_{t-1}^{(j)} K_t(\theta_{t-1}^{(j)}, \theta_t^{(i)})} & \text{if } t > 1 \end{cases}$$

4. Once all weights have been computed normalize them to 1, then advance to next iteration $t + 1$.

Notice how the particles interact: at iteration t each new particle is computed by choosing one from the previous population with *discrete* probabilities w_{t-1} (approximating π_{t-1}), then perturbed via K_t. This means no particles are actually sampled from scratch; we simply let the initial ones evolve so that their collective distribution gets closer to the target density. This allows for trivial parallelization: $\theta_t^{(i)}$ needs $\{\theta_{t-1}^{(j)}\}$ but not the rest of $\{\theta_t^{(i)}\} \implies$ we only need to sync populations at the beginning of t (cfr MCMC).

Recap: final *astroABC* algorithm

```
1: Set the tolerance thresholds, $\epsilon_t$ for $t = 0 \cdots T$ iterations.
2: procedure ABC SMC LOOP
3: At iteration t=0:
4:     for $1 \leq i \leq N$ do
5:         while $\rho(D, D^*) > \epsilon_0$ do
6:             Sample $\theta^*$ from prior $\theta^* \sim \pi(\theta)$
7:             Simulate mock data $D^* \sim \mathrm{M}(D|\theta^*)$
8:             Calculate distance metric $\rho(D, D^*)$
9:         Set $\theta_{i,0} \leftarrow \theta^*$
10:        Set weights $w_{i,0} \leftarrow 1/N$
11:    Set covariance $\Sigma_0^2 \leftarrow 2\Sigma(\theta_{1:N,0})$
12: At iteration $t > 0$:
13:    for $1 < t < T$ do
14:        for $1 \leq i \leq N$ do
15:            while $\rho(D, D^*) > \epsilon_t$ do
16:                Sample $\theta^*$ from previous iteration. $\theta^* \sim \theta_{1:N,t-1}$ with
    probabilities $w_{1:N,t-1}$
17:                Perturb $\theta^*$ by sampling $\theta^{**} \sim \mathcal{N}(\theta^*, \Sigma_{t-1}^2)$
18:                Simulate mock data $D^* \sim \mathrm{M}(D|\theta^{**})$
19:                Calculate distance metric $\rho(D, D^*)$
20:            Set $\theta_{i,t} \leftarrow \theta^{**}$
21:            Set weights $w_{i,t} \leftarrow \frac{\pi(\theta_{i,t})}{\sum_{j=1}^{N} w_{j,t-1}\mathcal{K}(\theta_{j,t-1}|\theta_{i,t},\Sigma_{t-1})}$ using kernel $\mathcal{K}$
22:        Set covariance $\Sigma_t^2$ using e.g. twice weighted empirical covariance
```

Table of Contents

astroABC's features I

- *Parallelization:* As noted before for fixed t each iteration the loop over particles can be executed in parallel. Each particles is resampled until we find a suitable candidate, and once the full population has been updated we can proceed to $t + 1$. This parallel-with-synchronization execution is efficiently implemented in astroABC using `MPI` for large jobs, and python's `multiprocessing` for small ones.

astroABC's features I

- *Parallelization:* As noted before for fixed t each iteration the loop over particles can be executed in parallel. Each particles is resampled until we find a suitable candidate, and once the full population has been updated we can proceed to $t+1$. This parallel-with-synchronization execution is efficiently implemented in astroABC using `MPI` for large jobs, and python's `multiprocessing` for small ones.
- *Priors:* astroABC allows the user to assign different prior distributions to each parameter, both standard ones e.g. (gaussians) and custom ones (e.g. implementing the results of previous analyses, which can be done for ex. by supplying previous Markov chains).

astroABC's features I

- *Parallelization:* As noted before for fixed t each iteration the loop over particles can be executed in parallel. Each particles is resampled until we find a suitable candidate, and once the full population has been updated we can proceed to $t + 1$. This parallel-with-synchronization execution is efficiently implemented in astroABC using `MPI` for large jobs, and python's `multiprocessing` for small ones.
- *Priors:* astroABC allows the user to assign different prior distributions to each parameter, both standard ones e.g. (gaussians) and custom ones (e.g. implementing the results of previous analyses, which can be done for ex. by supplying previous Markov chains).
- *Metric and summary statistics:* astroABC allows the user to define the sufficient statistics and distance metric used in the sampling.

astroABC's features II

- *ABC SMC:* astroABC allows the user to set at runtime one of a choice of tolerance thresholds, particle covariance estimators and perturbation kernels (more on this later).

astroABC's features II

- *ABC SMC:* astroABC allows the user to set at runtime one of a choice of tolerance thresholds, particle covariance estimators and perturbation kernels (more on this later).
- *Convenience features:* astroABC allows the user to frequently backup output files and restart files in case a job is canceled due to time restrictions.

astroABC's covariance estimators I

- In astroABC the distribution covariance is iteratively estimated/refined using the evolving population of particles (instead of computing it once and for all at the start via complex simulations, then fed e.g. in an MCMC-ready likelihood).

astroABC's covariance estimators I

- In astroABC the distribution covariance is iteratively estimated/refined using the evolving population of particles (instead of computing it once and for all at the start via complex simulations, then fed e.g. in an MCMC-ready likelihood).
- The covariance of accepted particles from the previous iteration is used in the SMC adaptive transition kernel K_t, which for particle $\theta_t^{(i)}$ is a gaussian kernel with mean $\theta_t^{(i)}$ and covariance equal to this matrix. The user also has the option to use a diagonal covariance matrix (i.e. neglecting correlations between parameters by removing off-diagonal elements) instead of the full matrix.

astroABC's covariance estimators II

To compute the entries in the covariance matrix the following options are available:

- twice the empirical covariance amongst the particles;
- covariances which result in a kernel which minimizes the Kullback-Leibler divergence between the target distribution and the distribution of the perturbed particles;
- local covariance estimate using scikit-learn's `KDTree` method for nearest neighbours;
- twice the weighted particle covariance matrix;
- a shrinkage covariance metric with the Ledoit-Wolf estimator.

Nontrivial options in the above list are motivated/discussed in the astroABC paper's references.

astroABC's tolerance level scheduler I

- As we know if ε is too large we recover the prior $\implies$ we learn nothing, whereas if ε is too small too many samples are rejected $\implies$ the algorithm is so picky it becomes too inefficient.

astroABC's tolerance level scheduler I

- As we know if ε is too large we recover the prior $\implies$ we learn nothing, whereas if ε is too small too many samples are rejected $\implies$ the algorithm is so picky it becomes too inefficient.
- Achieving a balance between these two cases can be hard; instead of finding a single good value for ε we can define a decreasing sequence $\{\varepsilon_t\}$.

astroABC's tolerance level scheduler I

- As we know if ε is too large we recover the prior $\implies$ we learn nothing, whereas if ε is too small too many samples are rejected $\implies$ the algorithm is so picky it becomes too inefficient.
- Achieving a balance between these two cases can be hard; instead of finding a single good value for ε we can define a decreasing sequence $\{\varepsilon_t\}$.
- At the beginning we haven't yet learned much, so in order to get closer to the posterior it makes sense to have a larger tolerance i.e. to accept even parameters that don't represent the observed data with high accuracy.

astroABC's tolerance level scheduler I

- As we know if ε is too large we recover the prior $\implies$ we learn nothing, whereas if ε is too small too many samples are rejected $\implies$ the algorithm is so picky it becomes too inefficient.
- Achieving a balance between these two cases can be hard; instead of finding a single good value for ε we can define a decreasing sequence $\{\varepsilon_t\}$.
- At the beginning we haven't yet learned much, so in order to get closer to the posterior it makes sense to have a larger tolerance i.e. to accept even parameters that don't represent the observed data with high accuracy.
- Similarly as the algorithm progresses we can afford lower efficiency because we already have an approximation of the posterior, and what we need is to improve it with high accuracy estimates. For this reason the tolerance level decreases over time, so that later on the parameters are sampling a high probability region of the posterior distribution.

astroABC's tolerance level scheduler II

- Within astroABC users can select a linearly decreasing, log decreasing, exponentially decreasing, constant or iteratively adaptive tolerance threshold.

astroABC's tolerance level scheduler II

- Within astroABC users can select a linearly decreasing, log decreasing, exponentially decreasing, constant or iteratively adaptive tolerance threshold.
- For the iteratively adaptive option the user specifies a certain quantile e.g. 75th, of the metric distance from the previous iteration. As such this decreasing tolerance depends on the particle positions in the previous iteration.

astroABC's tolerance level scheduler II

- Within astroABC users can select a linearly decreasing, log decreasing, exponentially decreasing, constant or iteratively adaptive tolerance threshold.
- For the iteratively adaptive option the user specifies a certain quantile e.g. 75th, of the metric distance from the previous iteration. As such this decreasing tolerance depends on the particle positions in the previous iteration.
- All tolerances require an input maximum and minimum value and the selected tolerance type is then implemented from a maximum value until either the minimum value is reached or the particles have reached the maximum number of iterations requested.

Table of Contents

astroABC paper's example

Citing directly from [1]: "In this section we present a simple example of parameter inference, with a simulated supernovae dataset, using both a standard MCMC sampler and astroABC. *This example represents a nontrivial case which has been specifically chosen to highlight potential biases in the MCMC approach which can be avoided using ABC.* Note this is not meant to show the best constraints we can get from ABC versus MCMC using supernova data. *The toy example is meant to breakdown key elements and assumptions about the two methods with a simple physical example* and is intended as a teaching exercise."
Similar situation as in the BHM SN Ia paper, but where we purposely "cheat" to make the BHM rigorous approach fail in a way that is automatically prevented by using ABC.

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following “true” values:

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following "true" values:
  - Matter density of the universe today: $\Omega_m = 0.3$

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following "true" values:
  - Matter density of the universe today: $\Omega_m = 0.3$
  - Dark energy density of the universe today: $\Omega_\Lambda = 0.7$

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following "true" values:
  - Matter density of the universe today: $\Omega_m = 0.3$
  - Dark energy density of the universe today: $\Omega_\Lambda = 0.7$
  - Present value of the dark energy equation of state: $w_0 = -1.0$

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following "true" values:
  - Matter density of the universe today: $\Omega_m = 0.3$
  - Dark energy density of the universe today: $\Omega_\Lambda = 0.7$
  - Present value of the dark energy equation of state: $w_0 = -1.0$
- At every redshift we add *non gaussian noise* by adding random variates from a skewed normal distribution with fixed parameters: location $= -0.1$, scale $= 0.3$, skew $= 5.0$; this is the crucial point, more on this in the next slide

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following “true” values:
  - Matter density of the universe today: $\Omega_m = 0.3$
  - Dark energy density of the universe today: $\Omega_\Lambda = 0.7$
  - Present value of the dark energy equation of state: $w_0 = -1.0$
- At every redshift we add *non gaussian noise* by adding random variates from a skewed normal distribution with fixed parameters: location $= -0.1$, scale $= 0.3$, skew $= 5.0$; this is the crucial point, more on this in the next slide
- During sampling we assume a flat universe and no prior information from other probes; we only assume wide gaussian priors on both parameters:

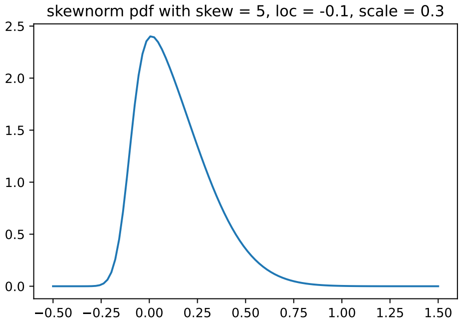
Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following "true" values:
    - Matter density of the universe today: $\Omega_m = 0.3$
    - Dark energy density of the universe today: $\Omega_\Lambda = 0.7$
    - Present value of the dark energy equation of state: $w_0 = -1.0$
- At every redshift we add *non gaussian noise* by adding random variates from a skewed normal distribution with fixed parameters: location $= -0.1$, scale $= 0.3$, skew $= 5.0$; this is the crucial point, more on this in the next slide
- During sampling we assume a flat universe and no prior information from other probes; we only assume wide gaussian priors on both parameters:
    - $\pi(\Omega_m) = \mathcal{N}(0.3, 0.5)$

Problem setup

- In this example a mock dataset of 400 supernovae in the $0.5 < z < 1.0$ redshift range is sampled using the following "true" values:
  - Matter density of the universe today: $\Omega_m = 0.3$
  - Dark energy density of the universe today: $\Omega_\Lambda = 0.7$
  - Present value of the dark energy equation of state: $w_0 = -1.0$
- At every redshift we add *non gaussian noise* by adding random variates from a skewed normal distribution with fixed parameters: location $= -0.1$, scale $= 0.3$, skew $= 5.0$; this is the crucial point, more on this in the next slide
- During sampling we assume a flat universe and no prior information from other probes; we only assume wide gaussian priors on both parameters:
  - $\pi(\Omega_m) = \mathcal{N}(0.3, 0.5)$
  - $\pi(w_0) = \mathcal{N}(-1, 0.5)$

Non-Gaussian noise I



- Adding this non-Gaussian random noise means that the final dataset has a non-Gaussian distribution at every redshift. In realistic datasets there may be several systematics which will have a similar effect e.g zero-point offset systematics in supernova studies.

Non-Gaussian noise II

- We assume that an analytic expression for the distribution of the final dataset is not available to us, but that e.g. non gaussian systematics can be accurately forward modeled using a simulation.

Non-Gaussian noise II

- We assume that an analytic expression for the distribution of the final dataset is not available to us, but that e.g. non gaussian systematics can be accurately forward modeled using a simulation.
- For this reason *in the context of this example* sticking to a Gaussian likelihood is inaccurate yet basically unavoidable within an MCMC approach; if instead we use ABC we may use the same physical model without also needing the full likelihood (more on this later).

Non-Gaussian noise II

- We assume that an analytic expression for the distribution of the final dataset is not available to us, but that e.g. non gaussian systematics can be accurately forward modeled using a simulation.
- For this reason *in the context of this example* sticking to a Gaussian likelihood is inaccurate yet basically unavoidable within an MCMC approach; if instead we use ABC we may use the same physical model without also needing the full likelihood (more on this later).

For now let us setup each model and see the results.

MCMC setup

In this toy example we assume a simple gaussian likelihood without realistic complications (like e.g. nonconstant absolute magnitude, which aren't present in the simple mock dataset):

$$\mathcal{L}(\mu_{\text{data}}|\mu_{\text{model}}(z,\Omega_m,w_0)) \propto \exp\left(-\sum_i \left(\frac{\mu^i_{\text{data}} - \mu_{\text{model}}(z^i,\Omega_m,w_0)}{2\sigma^i}\right)^2\right)$$

where μ^i_{data} is the distance modulus for an individual supernova in the data, with associated error σ^i, and where (flat universe assumption)

$$\mu_{\text{model}}(z^i,\Omega_m,w_0) \propto 5\log_{10}\frac{c(1+z)}{h_0}\int_0^{z^i} \mathrm{d}z' \, \frac{1}{E(z')}$$

$$E(z) = \sqrt{\Omega_m(1+z)^3 + (1-\Omega_m)\exp\left(3\int_0^z \mathrm{d}\ln(1+z')\,(1+w(z'))\right)}$$

MCMC results I

With this MCMC and 28 chains with 10^5 points each the results are the following:

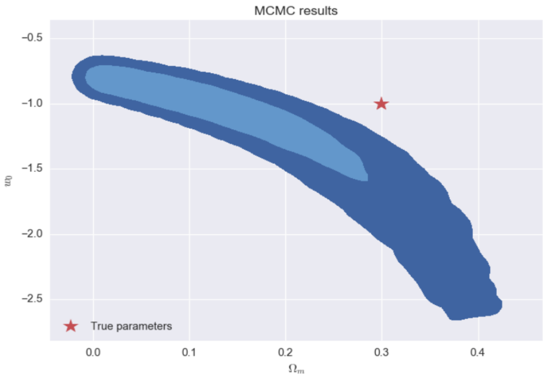


Figure: $P_{\text{MCMC}}(\Omega_m, w_0|D)$ with 1σ and 2σ contours shown.

MCMC results II

- The true parameters are clearly outside of the 2σ credibility region. This is also confirmed by the 1σ marginalized best fit values: $\Omega_m = 0.17 \pm 0.11$, $w_0 = -1.26 \pm 0.55$. Even though the "true" value -1.0 for w_0 is within its 1σ interval $\Omega_m = 0.3$ clearly isn't, as can be seen in the previous plot.

MCMC results II

- The true parameters are clearly outside of the 2σ credibility region. This is also confirmed by the 1σ marginalized best fit values: $\Omega_m = 0.17 \pm 0.11$, $w_0 = -1.26 \pm 0.55$. Even though the "true" value -1.0 for w_0 is within its 1σ interval $\Omega_m = 0.3$ clearly isn't, as can be seen in the previous plot.
- *It is clear that the simple Gaussian Likelihood assumption in this case (which neglects the effects of systematics) yields biased cosmological constraints.*

MCMC results II

- The true parameters are clearly outside of the 2σ credibility region. This is also confirmed by the 1σ marginalized best fit values: $\Omega_m = 0.17 \pm 0.11$, $w_0 = -1.26 \pm 0.55$. Even though the "true" value -1.0 for w_0 is within its 1σ interval $\Omega_m = 0.3$ clearly isn't, as can be seen in the previous plot.
- *It is clear that the simple Gaussian Likelihood assumption in this case (which neglects the effects of systematics) yields biased cosmological constraints.*
- We can do better with ABC in this specific instance.

ABC setup I

- In this toy example we assume that our noisy data can be simulated accurately and easily, but that an analytical expression for the likelihood is not available to us.

ABC setup I

- In this toy example we assume that our noisy data can be simulated accurately and easily, but that an analytical expression for the likelihood is not available to us.
- Using a forward model simulation we can account for non-Gaussian uncertainties in the data without explicitly knowing the likelihood. In particular we assume our simulation is able to draw gaussian random variates from the same $\mu_{\text{model}}(z^i, \Omega_m, w_0) = \dots$ model, *to which we add non-Gaussian noise from a skew normal distribution* (i.e. the systematics).

ABC setup I

- In this toy example we assume that our noisy data can be simulated accurately and easily, but that an analytical expression for the likelihood is not available to us.
- Using a forward model simulation we can account for non-Gaussian uncertainties in the data without explicitly knowing the likelihood. In particular we assume our simulation is able to draw gaussian random variates from the same $\mu_{\text{model}}(z^i, \Omega_m, w_0) = \ldots$ model, *to which we add non-Gaussian noise from a skew normal distribution* (i.e. the systematics).
- *Remark:* the use of exactly the same model here in the two methods is to highlight the distinction between choice of *model*, and choice of *Likelihood* in inference techniques. *Even though the physical model can be the same in both, an incorrect Likelihood assumption can bias results.*

ABC setup II

- Also notice that with ABC it is also possible to parametrize the source of non-Gaussian noise in the simulation and fit for e.g. the hyperparameters of the skew normal distribution, too.

ABC setup II

- Also notice that with ABC it is also possible to parametrize the source of non-Gaussian noise in the simulation and fit for e.g. the hyperparameters of the skew normal distribution, too.
- Using ABC at every iteration we simulate a set of supernovae at every point in a two dimensional parameter space $\{(\Omega_m, w_0)\}$; this simulated output is then compared with the dataset using a weighted Euclidean metric and an iteratively adaptive threshold.

ABC setup II

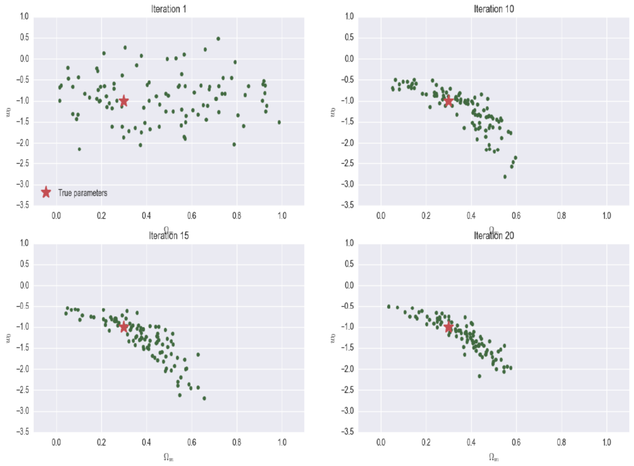
- Also notice that with ABC it is also possible to parametrize the source of non-Gaussian noise in the simulation and fit for e.g. the hyperparameters of the skew normal distribution, too.
- Using ABC at every iteration we simulate a set of supernovae at every point in a two dimensional parameter space $\{(\Omega_m, w_0)\}$; this simulated output is then compared with the dataset using a weighted Euclidean metric and an iteratively adaptive threshold.
- In the paper the authors use 100 particles and run until the error on the 1σ contour is $\sim 5\%$; this choice for the number of particles is based on trial and error runs to determine an optimal way to sample a multi dimensional parameter space efficiently.

ABC setup II

- Also notice that with ABC it is also possible to parametrize the source of non-Gaussian noise in the simulation and fit for e.g. the hyperparameters of the skew normal distribution, too.
- Using ABC at every iteration we simulate a set of supernovae at every point in a two dimensional parameter space $\{(\Omega_m, w_0)\}$; this simulated output is then compared with the dataset using a weighted Euclidean metric and an iteratively adaptive threshold.
- In the paper the authors use 100 particles and run until the error on the 1σ contour is $\sim 5\%$; this choice for the number of particles is based on trial and error runs to determine an optimal way to sample a multi dimensional parameter space efficiently.

First let us plot the progress of the ABC particles as a function of time compared to the true values (red star).

ABC results I



It is clear that at $t = 1$ the particles are well dispersed throughout the prior range. As the tolerance threshold level decreases the particles converge towards the "true" values, distributing like the posterior.

ABC results II

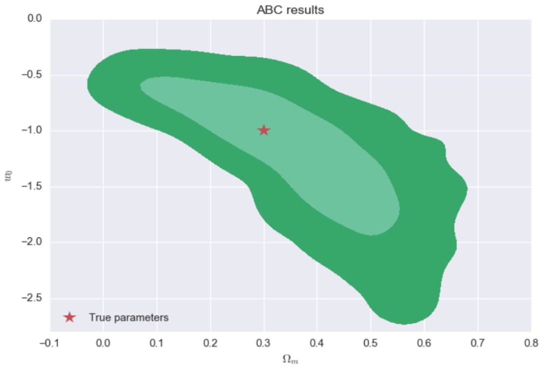


Figure: $P_{\text{ABC}}(\Omega_m, w_0|D)$ with 1σ and 2σ contours shown.

This time the posterior is clearly unbiased, and accurately centered on the "true" values.

ABC results III

- The 1σ marginalized best fit values are $\Omega_m = 0.36 \pm 0.12$, $w_0 = -1.22 \pm 0.4$ inside which the true values comfortably sit.

ABC results III

- The 1σ marginalized best fit values are $\Omega_m = 0.36 \pm 0.12$, $w_0 = -1.22 \pm 0.4$ inside which the true values comfortably sit.
- It's evident that with a forward simulation model one can naturally incorporate systematic effects at every point in parameter space, while avoiding the explicit calculation of the Likelihood and any parameter bias seen in the MCMC method.

Table of Contents

Conclusions

From the astroABC paper [1]: "We have described astroABC [. . .] [its] sampler can be used in parameter inference by simulating observations from posterior distributions when likelihoods are difficult or impossible to compute. Problems such as this arise frequently in cosmological applications, where [. . .] often [. . .] a physical model for the data can be simulated rapidly, and [. . .] [with] systematic uncertainties, but is sufficiently complicated that explicit formulae for the Likelihood are not known. The demand for alternative sampling methods in astronomy is increasing given the large correlated datasets we expect to analyze from future surveys. Current methods increasingly rely on simulations for error estimation and ABC sampling is a natural extension of this approach with the advantage that the Likelihood is not explicitly calculated. astroABC was designed to be [. . .] user friendly [. . .] while also accommodating the computational demands of simulating future datasets [. . .] with its facility for massive parallelization using MPI [. . .] and innovative factors such as a varied choice of covariance and kernel estimation, tolerance levels and priors."

Bibliography I

[1] E. Jennings and M. Madigan. "astroABC : An Approximate Bayesian Computation Sequential Monte Carlo sampler for cosmological parameter estimation". In: *Astronomy and Computing* 19 (Apr. 2017), pp. 16–22. DOI: 10.1016/j.ascom.2017.01.0012. URL: https://doi.org/10.1016%2Fj.ascom.2017.01.001.

[2] Umberto Picchini. *An intro to ABC – approximate Bayesian computation*. URL: https://www.maths.lu.se/fileadmin/maths/forskning_research/InferPartObsProcess/abc_slides.pdf.

[3] Tina Toni et al. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *Journal of The Royal Society Interface* 6.31 (July 2008), pp. 187–202. DOI: 10.1098/rsif.2008.0172. URL: https://doi.org/10.1098%2Frsif.2008.0172.

Bibliography II

[4] Wikipedia volunteers. *Conjugate priors*. URL: https://en.wikipedia.org/wiki/Conjugate_prior.