Model Calibration with Approximate Bayesian Computation

Nicolò Gozzi, ISI Foundation

2025/03/10 - 2025/03/12 NetSI

ISI Foundation Education

MOBS LAB

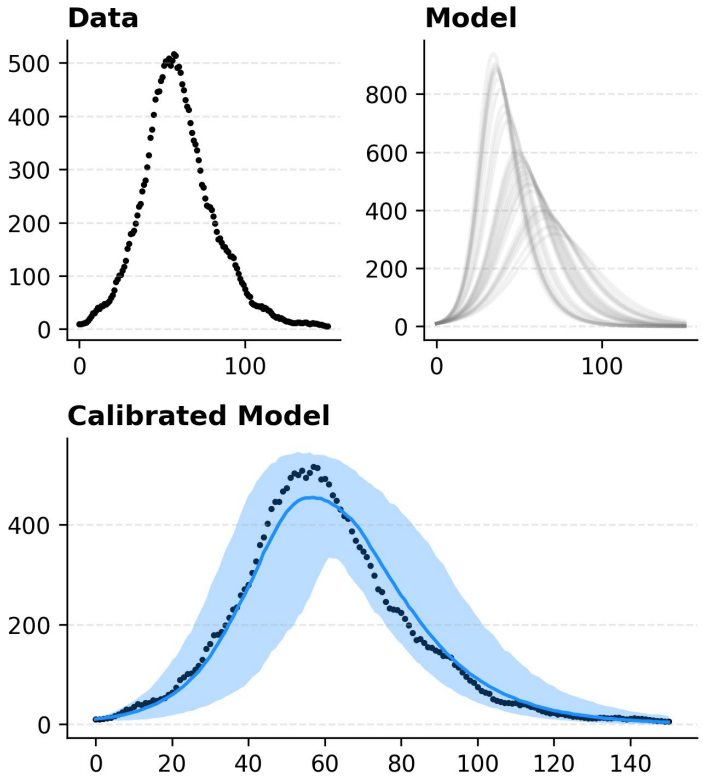
The problem: Model Calibration

You have:

- Some kind **data** (experimentally observed, synthetically generated, etc)
- A **parameterized simulator** (i.e., a function depending on some parameters which supposedly explains the data)

You want:

- To know which parameters well match the data and which ones not (i.e. **parameter inference**)



The problem: Model Calibration

- Some parameters can be informed by previous studies (e.g., what is the latent period for influenza?)
- Alternatively, values of (free) parameters can be optimized through statistical procedures to better reproduce data
- **Approximate Bayesian Computation** (ABC) techniques are a set of tools to calibrate models

Approximate Bayesian Computation

Approximate Bayesian **Computation**

Approximate Bayesian **Computation**

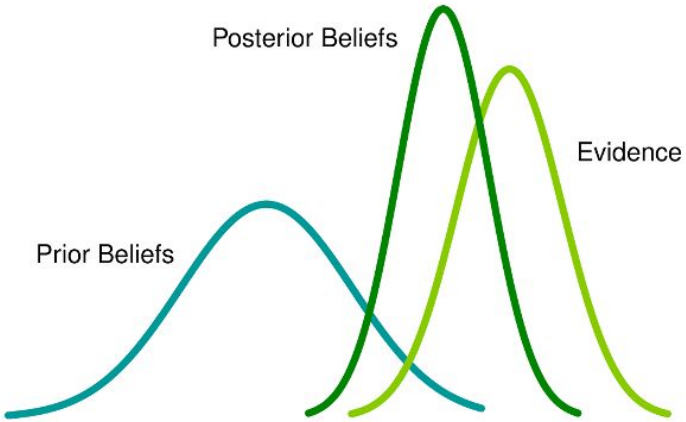
- Computational methods
- Focused on practical applications
- Easy to translate in algorithms

Approximate **Bayesian** Computation

What is Bayesian Statistics?

Bayesian statistics is a theory in the field of statistics based on the Bayesian interpretation of probability:

- The main concept of Bayesian inference is the idea of updating beliefs with new evidence (*Today's posterior is tomorrow's prior*)
- Within the Bayesian framework, parameters are treated as random variables characterized by probability distributions, rather than exact values



The Bayes Theorem

These ideas are condensed in the famous Bayes theorem, which is a rule for updating probabilities based on new evidence:

$$P(\theta|D) \propto P(\theta)P(D|\theta)$$

The Bayes Theorem

These ideas are condensed in the famous Bayes theorem, which is a rule for updating probabilities based on new evidence:

$$P(\theta|D) \propto \boxed{P(\theta)}P(D|\theta)$$

Prior distribution: What we believe before seeing data (This is what we decide)

The Bayes Theorem

These ideas are condensed in the famous Bayes theorem, which is a rule for updating probabilities based on new evidence:

$$P(\theta|D) \propto P(\theta)\boxed{P(D|\theta)}$$

Likelihood function: How likely the data is given the model (This is what we have to compute)

The Bayes Theorem

These ideas are condensed in the famous Bayes theorem, which is a rule for updating probabilities based on new evidence:

$$\boxed{P(\theta|D)} \propto P(\theta)P(D|\theta)$$

Posterior distribution: Updated belief after observing data
(This is what we want to obtain)

Bayesian vs Frequentist Approach

Two different philosophies in statistical inference:

	Bayesian Approach *(tl;dr: Parameters have distributions, data updates beliefs)*	**Frequentist Approach** *(tl;dr: Parameters are fixed, data is random)*
Parameters	Treated as random variables with probability distributions	Treated as fixed but unknown values
Probability	Represents degree of belief	Represents long-run frequency of events
Prior Knowledge	Incorporates prior beliefs, which are updated with data	No prior information, only data informs inference
Inference	Produces a posterior distribution for parameters	Relies on point estimates and confidence intervals

Bayesian vs Frequentist Approach

Two different philosophies in statistical inference:

	Bayesian Approach *(tl;dr: Parameters have distributions, data updates beliefs)*	**Frequentist Approach** *(tl;dr: Parameters are fixed, data is random)*
Parameters	Treated as random variables with probability distributions	Treated as fixed but unknown values
Probability	Represents degree of belief	Represents long-run frequency of events
Prior Knowledge	Incorporates prior beliefs, which are updated with data	No prior information, only data informs inference
Inference	Produces a posterior distribution for parameters	Relies on point estimates and confidence intervals

Despite philosophical disputes, Bayesian approaches have several advantages:

1. **Continuous beliefs refinement** using observed data
2. **Natural inclusion of uncertainty** (outputs are posterior distributions, not just point estimates)
3. **Naturally handles small data sets,** as priors help stabilize estimates

Approximate Bayesian Computation

Why do we need approximate methods?

- In practice, the Bayes theorem is rarely applied directly to get the expression of $P(\theta|D)$
- Indeed, apart from trivial cases, it is hard to get an analytical expression for the likelihood $P(D|\theta)$
- This is where Approximate Bayesian Computation (ABC) techniques come into play
- The goal of ABC is to estimate the posterior distribution of the parameters without computing the likelihood function

When to use and not to use ABC

- If you can explicitly define the likelihood, then it may be more effective to use an alternative inference technique that takes full advantage of its structure may be a more effective approach
- Likelihood-free inference such as ABC is powerful when you cannot analytically calculate the likelihood function.

Importance of ABC methods in Epidemiology, Ecology, and Complex Systems

Epidemiology - Infectious Disease Modeling

- Disease spread models (SIR, SEIR) often involve stochastic processes and heterogeneous populations
- Likelihood functions are intractable due to nonlinear dynamics and missing data

Ecology - Population & Evolutionary Dynamics

- Evolutionary and ecological models rely on genetic drift, migration, and selection—processes that are hard to express in a likelihood function.
- Observational data is often incomplete

Complex Systems - Agent-Based & Network Models

- Many systems (e.g., financial markets, climate systems, social behavior) involve many interacting agents, making traditional likelihood-based inference infeasible
- Models rely on simulations, not closed-form equations

Approximate Bayesian Computation for infectious disease modelling - ScienceDirect

ABC Algorithms

<u>Basic Ingredients</u> of an ABC algorithm

1. Prior distribution of free parameters $P(\theta)$
2. The data D
3. A distance function between the data and the model output: $d(D, D^*)$
4. A simulator $f(..., \theta)$

A simple ABC technique: the Rejection Algorithm

Idea:

- Iteratively sample from the prior distribution
- Accept/reject parameters based on the distance between their model output and the data

A simple ABC technique: the Rejection Algorithm

Idea:

- Iteratively sample from the prior distribution
- Accept/reject parameters based on the distance between their model output and the data

Algorithm 1 ABC Rejection Algorithm

Input:
- D_{obs}: Observed data
- $f(\cdot|\boldsymbol{\theta})$: Simulator model
- $P(\boldsymbol{\theta})$: Prior distribution
- $d(\cdot,\cdot)$: Distance function
- ϵ: Distance Tolerance
- P: Number of particles

Output: Posterior distribution approximations $\{\boldsymbol{\theta}_i\}_{i=1}^{P}$

Initialize accepted samples set: $\Theta = \emptyset$
while $|\Theta| < P$ **do**
  Sample $\theta^* \sim P(\theta)$ ▷ Draw from prior
  Simulate data $D^* \sim f(\cdot|\boldsymbol{\theta}^*)$ ▷ Run simulator with θ^*
  if $d(D_{obs}, D^*) < \epsilon$ **then** ▷ Check similarity
    Add θ^* to Θ ▷ Accept the sample
  end if
end while
return Θ

A simple ABC technique: the <u>Rejection</u> Algorithm

Overall idea: iteratively sample from the prior distribution and accept/reject parameters based on the distance between their model output and the data

Algorithm 1 ABC Rejection Algorithm

Input:
- D_{obs}: Observed data
- $f(\cdot|\boldsymbol{\theta})$: Simulator model
- $P(\boldsymbol{\theta})$: Prior distribution
- $d(\cdot,\cdot)$: Distance function
- ϵ: Distance Tolerance
- P: Number of particles

Output: Posterior distribution approximations $\{\boldsymbol{\theta}_i\}_{i=1}^{P}$

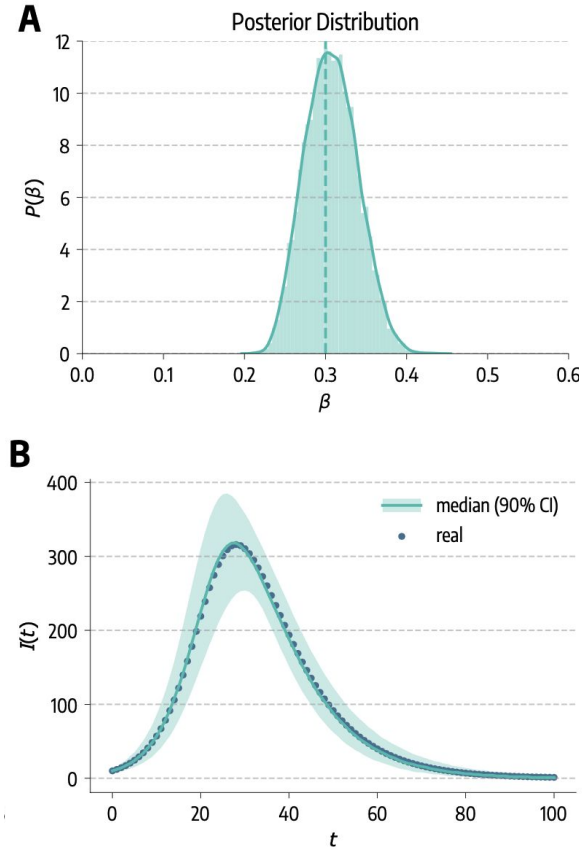
Initialize accepted samples set: $\Theta = \emptyset$
while $|\Theta| < P$ **do**
  Sample $\theta^* \sim P(\theta)$ ▷ Draw from prior
  Simulate data $D^* \sim f(\cdot|\boldsymbol{\theta}^*)$ ▷ Run simulator with θ^*
  if $d(D_{obs}, D^*) < \epsilon$ **then** ▷ Check similarity
    Add θ^* to Θ ▷ Accept the sample
  end if

The P accepted samples are an approximation of the actual posterior distribution of model parameters

ABC Rejection Algorithm: Practical Example

We want to find the posterior distribution of transmission rate β using a stochastic SIR model

1. We set a uniform prior on the only free parameter β ∼U(0.01,0.6)
2. Other parameters are fixed to μ = 0.1, total number of individuals N = 1000, initial fraction of infected 1%
3. As distance metric, we use the weighted mean absolute percentage error (wMAPE) on the number of infected at each time step t
4. We set a tolerance of ε = 0.25 and we run the ABC-rejection algorithm to accept 1,000 particles



ABC Rejection Algorithm: Python Code

```python
import numpy as np

def abc_rejection(model_simulator, prior_sampler, distance_function,
                  observed_data, epsilon, num_particles):
    """Implements the ABC-Rejection algorithm"""
    accepted_params = []

    while len(accepted_params) < num_particles:
        # Sample from prior
        theta = prior_sampler()

        # Simulate data from model
        simulated_data = model_simulator(theta)

        # Compute distance
        distance = distance_function(simulated_data, observed_data)

        # Accept if distance is within tolerance
        if distance < epsilon:
            accepted_params.append(theta)

    return np.array(accepted_params)
```

Limitations of ABC Rejection Algorithm

- **Lack of adaptive learning:** information from previous iterations is never used (i.e., prior is never updated)
- **Curse of dimensionality**: becomes impractical for high-dimensional parameter spaces due to low acceptance rates.
- **Threshold sensitivity**: what is a good distance tolerance?
  - *High* tolerance: high acceptance rate, fast convergence, low accuracy
  - *Low* tolerance: low acceptance rate, slow convergence, high accuracy

Simulation-budget-constrained Rejection

Idea:

- Instead of setting a tolerance, set a maximum number of simulations
- Select the top X% simulations/parameters based on the error metric

Simulation-budget-constrained Rejection

Idea:

- Instead of setting a tolerance, set a maximum number of simulations
- Select the top X% simulations/parameters based on the error metric

Algorithm 2 Modified ABC Rejection Algorithm with Simulation Budget

Input:
  D_{obs}: Observed data
  $f(\cdot|\boldsymbol{\theta})$: Simulator model
  $P(\boldsymbol{\theta})$: Prior distribution
  $d(\cdot,\cdot)$: Distance function
  B: Total simulation budget (number of total simulations allowed)
  X: Percentage of best trajectories to select
Output: Selected parameter sets $\{\boldsymbol{\theta}_i\}_{i=1}^{\lfloor B \cdot X/100 \rfloor}$

Initialize an empty set of candidates: Candidates $= \emptyset$
for $i = 1$ to B **do**
  Sample $\boldsymbol{\theta}_i \sim P(\boldsymbol{\theta})$ ▷ Draw from prior
  Simulate data $D^* \sim f(\cdot|\boldsymbol{\theta}_i)$ ▷ Run simulator
  Compute distance $d(D^*, D_{\text{obs}})$ ▷ Assess similarity
  Add $(\boldsymbol{\theta}_i, d(D^*, D_{\text{obs}}))$ to Candidates
end for
Sort Candidates by $d(D^*, D_{\text{obs}})$ in ascending order ▷ Sort by similarity
Select the top $\lfloor B \cdot X/100 \rfloor$ parameter sets ▷ Retain best fits
return Selected parameter sets

The ABC-SMC Algorithm

Idea:

- The ABC-SMC algorithm solves these issues (SMC: Sequential Monte Carlo) by implementing iteratively the rejection algorithm
- Start from a high tolerance and refine the parameter space in next generations

Toni T, Welch D, Strelkowa N, Ipsen A, Stumpf MP. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. J R Soc Interface. 2009

The ABC-SMC Algorithm

Idea:

- The ABC-SMC algorithm solves these issues (SMC: Sequential Monte Carlo) by implementing iteratively the rejection algorithm
- Start from a high tolerance and refine the parameter space in next generations

Algorithm 3 ABC-SMC Algorithm for Parameters' Inference

Input:
- D_{obs}: Observed data
- $f(\cdot|\boldsymbol{\theta})$: Simulator model
- $P(\boldsymbol{\theta})$: Prior distribution
- $d(\cdot,\cdot)$: Distance function
- $\{\epsilon_t\}_{t=1}^{T}$: Sequence of tolerance levels
- P: Number of particles

Output: Posterior distribution approximations $\{\boldsymbol{\theta}_i^T\}_{i=1}^{P}$

Initialize population $\Theta^0 = \{\boldsymbol{\theta}_i^0\}_{i=1}^{P}$ by sampling $\boldsymbol{\theta}_i^0 \sim P(\boldsymbol{\theta})$
Set weights $w_i^0 = \frac{1}{P}$ for all i
for $t = 1, \ldots, T$ **do** ▷ Iterate over generations
  Initialize accepted samples set: $\Theta^t = \emptyset$
  while $|\Theta^t| < P$ **do**
    Sample $\boldsymbol{\theta}^*$ from previous population Θ^{t-1} with weights $\{w_i^{t-1}\}$
    Perturb $\boldsymbol{\theta}^*$ using a kernel $K_t(\cdot|\boldsymbol{\theta}^*)$
    Simulate data $D^* \sim f(\cdot|\boldsymbol{\theta}^*)$
    if $d(D_{\text{obs}}, D^*) < \epsilon_t$ **then** ▷ Check similarity
      Compute new weight:

$$w_i^* = \frac{P(\boldsymbol{\theta}^*)}{\sum_{j=1}^{P} w_j^{t-1} K_t(\boldsymbol{\theta}^*|\boldsymbol{\theta}_j^{t-1})}$$

      Add $\boldsymbol{\theta}^*$ to Θ^t with weight w_i^*
    end if
  end while
  Normalize weights: $w_i^t = \frac{w_i^*}{\sum_{j=1}^{P} w_j^*}$
end for
return Final population Θ^T with weights $\{w_i^T\}_{i=1}^{P}$

Toni T, Welch D, Strelkowa N, Ipsen A, Stumpf MP. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. J R Soc Interface. 2009

Why Perturbation Kernels?

Perturbation kernels are used to ensure effective exploration of the parameter space while maintaining diversity in accepted samples

- **Avoid particle degeneracy**: without perturbation, resampling from previous populations would lead to repeated samples, reducing diversity.
- **Prevent sample collapse:** if only the best-fitting samples were retained without perturbation, the algorithm could prematurely converge to a narrow region of parameter space.

On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo

How to choose perturbation kernels?

Trade-off:

- A wide kernel encourages more exploration (risk of inefficient search)
- A narrow kernel exploits local solutions but may miss better ones

Common Kernel choices:

- Gaussian Kernels
- Uniform Kernels
- **Adaptive Kernels**: Adjust kernel bandwidth based on sample variance to improve efficiency

On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo

Understanding Weights

Weights correct for changes in sampling probability between steps

$$w_i^* = \frac{\boxed{P(\boldsymbol{\theta}^*)}}{\sum_{j=1}^{P} w_j^{t-1} K_t(\boldsymbol{\theta}^* | \boldsymbol{\theta}_j^{t-1})}$$

Prior distribution (ensures consistency with the original prior belief)

Understanding Weights

Weights correct for changes in sampling probability between steps

$$w_i^* = \frac{P(\boldsymbol{\theta}^*)}{\sum_{j=1}^{P} w_j^{t-1} K_t(\boldsymbol{\theta}^* | \boldsymbol{\theta}_j^{t-1})}$$

Prior distribution (ensures consistency with the original prior belief)

A weighted sum of kernel-based proposals from the previous generation:

- This measures how likely the new sample could have been produced by perturbing previous samples.
- If it was very easy to generate from the previous generation its weight is downscaled because it was already well represented (and vice versa)

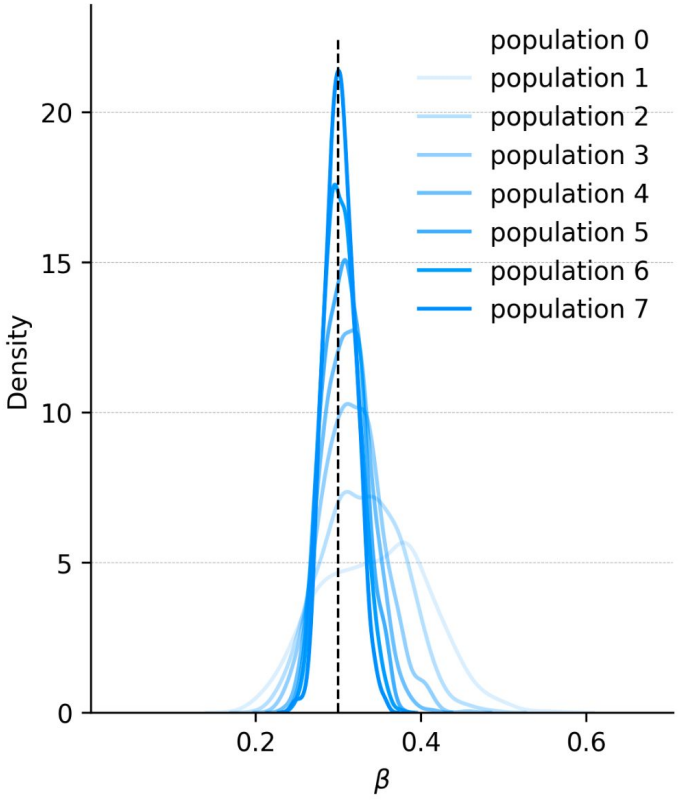
Choice of sequence of tolerances

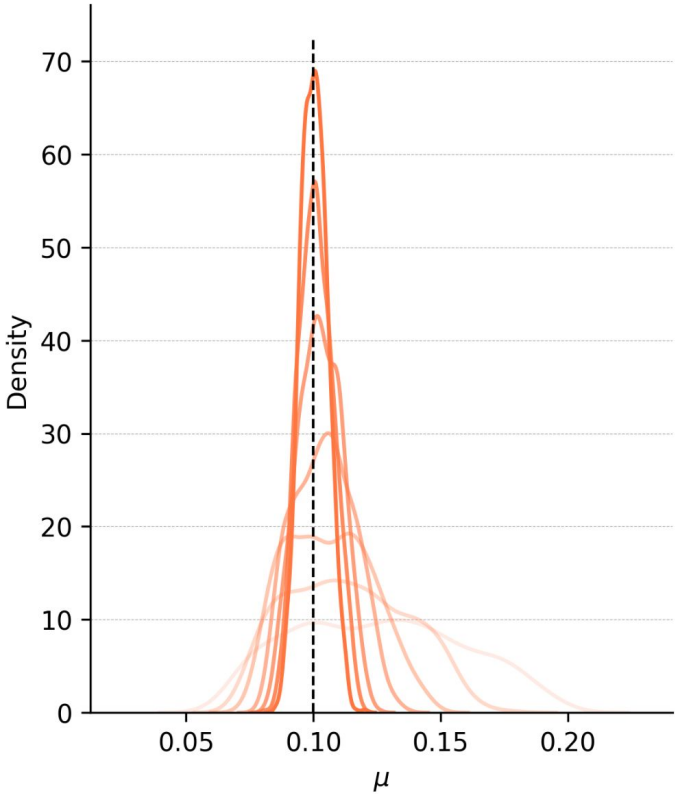
Overall idea: start from high tolerance and decrease. But how?

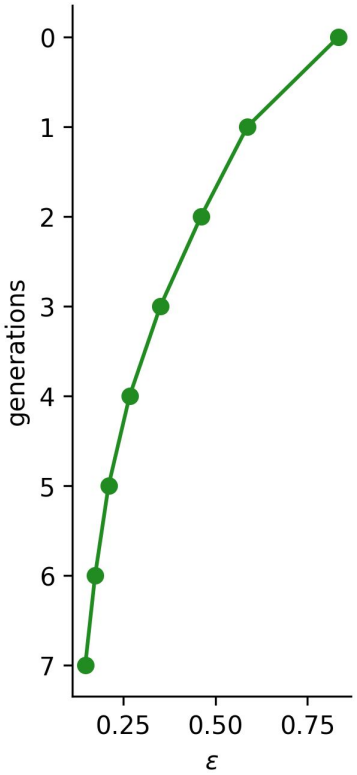
- Predefined schedule
- Fixed Decrease: $\varepsilon_t = \varepsilon_{t-1} \times c$, with $c<1$
- Quantile-Based Approach (**Adaptive Tolerance**):
  - Set ε_t as the X_{th} percentile of distances in the previous generation

An adaptive sequential Monte Carlo method for approximate Bayesian computation
Estimation of parameters for macroparasite population evolution using approximate bayesian computation - PubMed

ABC-SMC: Example







ABC-SMC for Model Selection

ABC-SMC can be slightly modified for model selection:

- When we have multiple competing models M_1, M_2, …, M_K
- Not only we want to infer parameters, but also the best models

Algorithm 4 ABC-SMC for Model Selection

Input:
- D_{obs}: Observed data
- $\{\mathcal{M}_k\}_{k=1}^K$: Set of candidate models
- $P(\mathcal{M})$: Prior distribution over models
- $P(\boldsymbol{\theta}|\mathcal{M})$: Prior distribution over parameters for each model
- $f(\cdot|\boldsymbol{\theta}, \mathcal{M})$: Simulator model
- $d(\cdot,\cdot)$: Distance function
- $\{\epsilon_t\}_{t=1}^T$: Sequence of tolerances
- P: Number of particles

Output: Approximate posterior probabilities $P(\mathcal{M}_k|D_{\text{obs}})$ and posterior samples $\{\boldsymbol{\theta}_i^T, \mathcal{M}_i^T\}_{i=1}^P$

Initialize model-parameter population $\Theta^0 = \{(\boldsymbol{\theta}_i^0, \mathcal{M}_i^0)\}_{i=1}^P$ by sampling:
- $\mathcal{M}_i^0 \sim P(\mathcal{M})$ ▷ Sample model from prior
- $\boldsymbol{\theta}_i^0 \sim P(\boldsymbol{\theta}|\mathcal{M}_i^0)$ ▷ Sample parameters from prior

Set weights $w_i^0 = \frac{1}{P}$ for all i

for $t = 1, \ldots, T$ **do** ▷ Iterate over generations
- Initialize accepted population: $\Theta^t = \emptyset$
- **while** $|\Theta^t| < P$ **do**
  - Sample $(\boldsymbol{\theta}^*, \mathcal{M}^*)$ from previous population Θ^{t-1} with weights $\{w_i^{t-1}\}$
  - Perturb $\boldsymbol{\theta}^*$ using a kernel $K_t(\cdot|\boldsymbol{\theta}^*)$
  - Simulate data $D^* \sim f(\cdot|\boldsymbol{\theta}^*, \mathcal{M}^*)$
  - **if** $d(D_{\text{obs}}, D^*) < \epsilon_t$ **then** ▷ Check similarity
    - Compute new weight:

$$w_i^* = \frac{P(\boldsymbol{\theta}^*|\mathcal{M}^*)P(\mathcal{M}^*)}{\sum_{j=1}^P w_j^{t-1} K_t(\boldsymbol{\theta}^*|\boldsymbol{\theta}_j^{t-1}) \mathbb{1}(\mathcal{M}_j^{t-1} = \mathcal{M}^*)}$$

    - Add $(\boldsymbol{\theta}^*, \mathcal{M}^*)$ to Θ^t with weight w_i^*
  - **end if**
- **end while**
- Normalize weights: $w_i^t = \frac{w_i^*}{\sum_{j=1}^P w_j^*}$

end for

Compute model posterior probabilities:

$$P(\mathcal{M}_k|D_{\text{obs}}) \approx \frac{\sum_{i=1}^P w_i^T \mathbb{1}(\mathcal{M}_i^T = \mathcal{M}_k)}{\sum_{i=1}^P w_i^T}$$

return Final population Θ^T and model probabilities $P(\mathcal{M}_k|D_{\text{obs}})$

Toni T, Welch D, Strelkowa N, Ipsen A, Stumpf MP. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. J R Soc Interface. 2009

ABC-SMC limitations

Distance Function Sensitivity:

- The choice of distance function affects inference quality
- Non-sufficient summaries lead to loss of information and bias

Curse of Dimensionality:

- ABC struggles with high-dimensional parameter spaces
- As dimensions increase, the probability of finding good matches decreases exponentially

Comparison with Other Calibration Methods (MLE)

	Maximum Likelihood Estimation (MLE)	**Approximate Bayesian Computation (ABC)**
Approach	Finds parameter θ^* that maximizes likelihood	Uses simulation-based likelihood-free inference
Likelihood Required?	Requires explicit likelihood function	No need for explicit likelihood
Computational Efficiency	Often faster if likelihood is available	Can be computationally expensive due to simulations
Uncertainty Quantification	Provides point estimates, not full distributions	Outputs full posterior distribution
Handling Complex Models	Difficult when likelihood is intractable	Works well for intractable likelihoods

Comparison with Other Calibration Methods (MCMC)

	Markov Chain Monte Carlo (MCMC)	**Approximate Bayesian Computation (ABC)**
Approach	Uses likelihood-based Bayesian inference with Metropolis-Hastings sampling	Uses simulation-based likelihood-free inference
Likelihood Required?	Requires explicit likelihood function	No need for explicit likelihood
Computational Cost	More efficient when likelihood is available	Can be computationally expensive due to simulations
Posterior Approximation	Exact (given sufficient samples)	Approximate, depends on ε and summary statistics
Scalability	May struggle with high-dimensional spaces	Also struggles, but ABC-SMC improves efficiency

Computational Tools: Epydemix

Epydemix is an open-source Python package for epidemiological modeling:

- Enables development and simulation of epidemic models
- Parameter inference with ABC methods
- Access to a wealth of real-world data on population and contacts

Pros: one tool to build and calibrate your model (even though works with any model, also external)

Cons: does not support (atm) multi-core sampling (i.e., large scale simulations and calibrations)

https://github.com/ngozzi/epydemix

epydemix
The ABC of epidemics

Computational Tools: PyABC

PyABC is an open-source Python package for performing likelihood-free Bayesian inference using ABC methods:

- It is designed for flexible, scalable, and efficient inference in complex models.
- intuitive API

Pros: support for multicore, large scale parallelized calibration, and model selection

Cons: general purpose library, not meant for epidemic modeling

https://pyabc.readthedocs.io/en/latest/

References

- Toni T, Welch D, Strelkowa N, Ipsen A, Stumpf MP. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. J R Soc Interface. 2009
- Approximate Bayesian Computation for infectious disease modelling - ScienceDirect
- Approximate Bayesian Computation | PLOS Computational Biology
- On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo