Accelerating ABC using history matching and GP emulators

Richard Wilkinson

School of Mathematical Sciences
University of Nottingham

r.d.wilkinson@nottingham.ac.uk

Southampton - April 2013

Thoughts

- Collaborate with Ian and Michael on comparing these things
- Ian suggests Darren Wilkinson's models (from front cover of his book), can be made to run for as long as you like.
- Suggest using 5*sd rather than 3 times and subtracting 10 for the threshold.

Southampton

- Talk is currently too long. Really is two talks. Cut something.
- Could add points to design one at a time. Could use some optimality criterion to place new points.

Talk Plan

1. Introduction - the need for simulation based methods
2. (Generalised) ABC as continuation of the modelling process
3. Using Gaussian processes to accelerate ABC algorithms

The need for simulation based methods

Rohrlich (1991): Computer simulation is

> *'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'*

The need for simulation based methods

Rohrlich (1991): Computer simulation is

> *'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'*

Challenges for statistics:
How do we make inferences about the world from a simulation of it?

The need for simulation based methods

Rohrlich (1991): Computer simulation is

> *'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'*

Challenges for statistics:
How do we make inferences about the world from a simulation of it?

- how do we relate simulators to reality? (model error)
- how do we estimate tunable parameters? (calibration)
- how do we deal with computational constraints? (stat. comp.)
- how do we make uncertainty statements about the world that combine models, data and their corresponding errors? (UQ)

There is an inherent a lack of quantitative information on the uncertainty surrounding a simulation - unlike in physical experiments.

Calibration

Focus on simulator calibration:

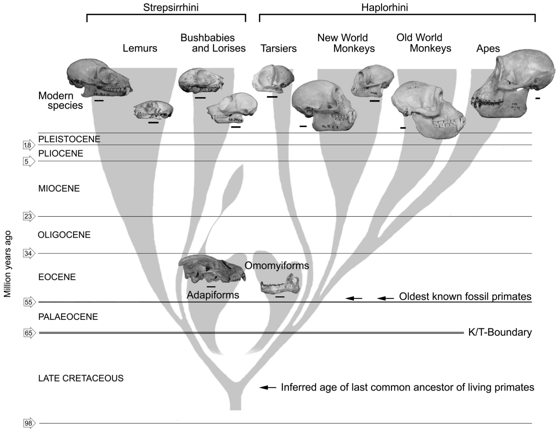
- For most simulators we specify parameters θ and i.c.s and the simulator, $f(\theta)$, generates output X.
- We are interested in the inverse-problem, i.e., observe data D, want to estimate parameter values θ that explain this data.

For Bayesians, this is a question of finding the posterior distribution

$$\pi(\theta|\mathcal{D}) \propto \pi(\theta)\pi(\mathcal{D}|\theta)$$

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

The likelihood isn't just the simulator pdf



Intractability

$$\pi(\theta|D) = \frac{\pi(D|\theta)\pi(\theta)}{\pi(D)}$$

A Bayesian inference problem is intractable if

$$\pi(D|\theta)$$

is unknown and can't be evaluated (unknown is subjective). I.e., if the analytic distribution of the simulator, $f(\theta)$, run at θ is unknown.
- worse than the usual normalising constant intractability

Intractability

$$\pi(\theta|D) = \frac{\pi(D|\theta)\pi(\theta)}{\pi(D)}$$

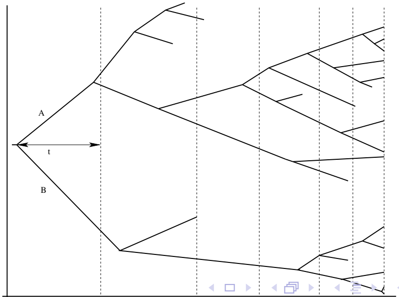
A Bayesian inference problem is intractable if

$$\pi(D|\theta)$$

is unknown and can't be evaluated (unknown is subjective). I.e., if the analytic distribution of the simulator, $f(\theta)$, run at θ is unknown.
- worse than the usual normalising constant intractability

Example:

The density of the cumulative process of a branching process is unknown in general. We could probably impute everything, but this will be costly.



Approximate Bayesian Computation (ABC)

Approximate Bayesian Computation (ABC)

ABC algorithms are a collection of Monte Carlo methods used for calibrating simulators

- they do not require explicit knowledge of the likelihood function $\pi(x|\theta)$
- inference is done using simulation from the model (they are 'likelihood-free').

ABC methods have become popular in the biological sciences and versions of the algorithm exist in most modelling communities.

Approximate Bayesian Computation (ABC)

ABC algorithms are a collection of Monte Carlo methods used for calibrating simulators

- they do not require explicit knowledge of the likelihood function $\pi(x|\theta)$
- inference is done using simulation from the model (they are 'likelihood-free').

ABC methods have become popular in the biological sciences and versions of the algorithm exist in most modelling communities.
ABC methods can be crude but they have an important role to play.

- Scientists are building simulators (intractable ones), and fitting them to data.
  - There is a need for simple methods that can be credibly applied.
  - Likelihood methods for complex simulators are complex.
  - Modelling is something that can be done well by scientists not trained in complex statistical methods.

Likelihood-Free Inference

Rejection Algorithm

- Draw θ from prior $\pi(\cdot)$
- Accept θ with probability $\pi(\mathcal{D} \mid \theta)$

Accepted θ are independent draws from the posterior distribution, $\pi(\theta \mid \mathcal{D})$.

Likelihood-Free Inference

Rejection Algorithm

- Draw θ from prior $\pi(\cdot)$
- Accept θ with probability $\pi(\mathcal{D} \mid \theta)$

Accepted θ are independent draws from the posterior distribution, $\pi(\theta \mid \mathcal{D})$.
If the likelihood, $\pi(\mathcal{D}|\theta)$, is unknown:

'Mechanical' Rejection Algorithm

- Draw θ from $\pi(\cdot)$
- Simulate $X \sim f(\theta)$ from the computer model
- Accept θ if $\mathcal{D} = X$, i.e., if computer output equals observation

The acceptance rate is $\mathbb{P}(\mathcal{D})$: the number of runs to get n observations is negative binomial, with mean $\frac{n}{\mathbb{P}(\mathcal{D})}$: $\Rightarrow$ Bayes Factors!

Uniform ABC algorithms

Uniform ABC

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(\mathcal{D}, X) \leq \epsilon$

For reasons that will become clear later, call this *Uniform ABC*.

Uniform ABC algorithms

Uniform ABC

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(\mathcal{D}, X) \leq \epsilon$

For reasons that will become clear later, call this *Uniform ABC*.

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid \mathcal{D})$

ϵ reflects the tension between computability and accuracy.

The hope is that $\pi_{ABC}(\theta) \approx \pi(\theta | D, PSH)$ for ϵ small, where PSH='perfect simulator hypothesis'

Uniform ABC algorithms

Uniform ABC

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(\mathcal{D}, X) \leq \epsilon$

For reasons that will become clear later, call this *Uniform ABC*.

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid \mathcal{D})$

ϵ reflects the tension between computability and accuracy.

The hope is that $\pi_{ABC}(\theta) \approx \pi(\theta|D, PSH)$ for ϵ small, where PSH='perfect simulator hypothesis'
There are uniform ABC-MCMC, ABC-SMC, ABC-EM, ABC-EP, ABC-MLE algorithms, etc.

Two ways of thinking

We think about linear regression in two ways

- Algorithmically: find the straight line that minimizes the sum of the squared errors.
- Probability model: we have a linear model with Gaussian errors, and we estimate the parameters using maximum-likelihood.

Two ways of thinking

We think about linear regression in two ways

- Algorithmically: find the straight line that minimizes the sum of the squared errors.
- Probability model: we have a linear model with Gaussian errors, and we estimate the parameters using maximum-likelihood.

We think about the Kalman filter in two ways:

- Algorithmically: linear quadratic estimation - find the best guess at the trajectory using linear dynamics and a quadratic penalty function
- Probability model: the (Bayesian) solution to the linear Gaussian filtering problem.

Two ways of thinking

We think about linear regression in two ways

- Algorithmically: find the straight line that minimizes the sum of the squared errors.
- Probability model: we have a linear model with Gaussian errors, and we estimate the parameters using maximum-likelihood.

We think about the Kalman filter in two ways:

- Algorithmically: linear quadratic estimation - find the best guess at the trajectory using linear dynamics and a quadratic penalty function
- Probability model: the (Bayesian) solution to the linear Gaussian filtering problem.

The same dichotomy exists for ABC.

Algorithmic view of ABC

Most of the early ABC developments have been in the algorithmic tradition.

1. Find a good metric, ρ - e.g., L_2 norm
2. Find a good ϵ - e.g., best 1% of simulations?
3. Find a good summary $S(D)$

The choices made are usually not motivated by modelling considerations.

Poor choices for any of these aspects can have unintended consequences.

Calibration framework

Lets now consider the probabilistic interpretation of ABC.

The Bayesian calibration framework from the computer experiment literature:

- Relate the best-simulator run ($X = f(\hat{\theta})$) to reality ζ
- Relate reality ζ to the observations D.

$$\begin{tikzcd}
A \arrow[r] & B \arrow[rr] & & C \arrow[rr] & & D \\
& & E \arrow[ur] & & F \arrow[ur] &
\end{tikzcd}$$

See, for example, Kennedy and O'Hagan (2001) or Goldstein and Rougier (2009).

Calibration framework

Mathematically, we can write the likelihood as

$$\pi(D|\theta) = \int \pi(D|x)\pi(x|\theta)\mathrm{dx}$$

where

- $\pi(D|x)$ is a pdf relating the simulator output to reality - call it the *acceptance kernel*.
- $\pi(x|\theta)$ is the likelihood function of the simulator (ie not relating to reality)

The posterior is

$$\pi(\theta|D) = \frac{1}{Z}\int \pi(D|x)\pi(x|\theta)\mathrm{dx}.\ \pi(\theta)$$

where $Z = \int\int \pi(D|x)\pi(x|\theta)\mathrm{dx}\pi(\theta)\mathrm{d}\theta$

Calibration framework

Mathematically, we can write the likelihood as

$$\pi(D|\theta) = \int \pi(D|x)\pi(x|\theta)\mathrm{dx}$$

where

- $\pi(D|x)$ is a pdf relating the simulator output to reality - call it the *acceptance kernel*.
- $\pi(x|\theta)$ is the likelihood function of the simulator (ie not relating to reality)

The posterior is

$$\pi(\theta|D) = \frac{1}{Z}\int \pi(D|x)\pi(x|\theta)\mathrm{dx}.\ \pi(\theta)$$

where $Z = \iint \pi(D|x)\pi(x|\theta)\mathrm{dx}\pi(\theta)\mathrm{d}\theta$

To simplify matters, we can work in joint (θ, x) space

$$\pi(\theta, x|D) = \frac{\pi(D|x)\pi(x|\theta)\pi(\theta)}{Z}$$

How does ABC relate to calibration?

Wilkinson 2008/2013

Consider how this relates to ABC:

$$\pi_{ABC}(\theta, x) := \pi(\theta, x|D) = \frac{\pi(D|x)\pi(x|\theta)\pi(\theta)}{Z}$$

Lets sample from this using the rejection algorithm with instrumental distribution

$$g(\theta, x) = \pi(x|\theta)\pi(\theta)$$

Generalized ABC (GABC)

Wilkinson 2008, Fearnhead and Prangle 2012

The rejection algorithm then becomes

Generalized rejection ABC (Rej-GABC)

1. $\theta \sim \pi(\theta)$ and $X \sim \pi(x|\theta)$ (ie $(\theta, X) \sim g(\cdot)$)
2. Accept (θ, X) if

$$U \sim U[0,1] \leq \frac{\pi_{ABC}(\theta, x)}{Mg(\theta, x)} = \frac{\pi(D|X)}{\max_x \pi(D|x)}$$

Generalized ABC (GABC)

Wilkinson 2008, Fearnhead and Prangle 2012

The rejection algorithm then becomes

Generalized rejection ABC (Rej-GABC)

1. $\theta \sim \pi(\theta)$ and $X \sim \pi(x|\theta)$ (ie $(\theta, X) \sim g(\cdot)$)
2. Accept (θ, X) if

$$U \sim U[0,1] \leq \frac{\pi_{ABC}(\theta, x)}{Mg(\theta, x)} = \frac{\pi(D|X)}{\max_x \pi(D|x)}$$

In uniform ABC we take

$$\pi(D|X) = \begin{cases} 1 & \text{if } \rho(D, X) \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

this reduces the algorithm to

2' Accept θ ifF $\rho(D, X) \leq \epsilon$

ie, we recover the *uniform* ABC algorithm.

Uniform ABC algorithm

This allows us to interpret uniform ABC. Suppose $X, D \in \mathcal{R}$

Proposition

Accepted θ from the uniform ABC algorithm (with $\rho(D, X) = |D - X|$) are samples from the posterior distribution of θ given D where we assume $D = f(\theta) + e$ and that

$$e \sim U[-\epsilon, \epsilon]$$

In general, uniform ABC assumes that

$$D|x \sim U\{d : \rho(d, x) \leq \epsilon\}$$

We can think of this as assuming a uniform error term when we relate the simulator to the observations.

Uniform ABC algorithm

This allows us to interpret uniform ABC. Suppose $X, D \in \mathcal{R}$

Proposition

Accepted θ from the uniform ABC algorithm (with $\rho(D, X) = |D - X|$) are samples from the posterior distribution of θ given D where we assume $D = f(\theta) + e$ and that

$$e \sim U[-\epsilon, \epsilon]$$

In general, uniform ABC assumes that

$$D|x \sim U\{d : \rho(d, x) \leq \epsilon\}$$

We can think of this as assuming a uniform error term when we relate the simulator to the observations.

ABC gives 'exact' inference under a different model!

Acceptance kernel $\pi(D|X)$ as an extension of modelling

Using ABC is equivalent to adding additional variability into the model.

- $\exists$ many interesting papers saying how to make this variability small
- Instead ask, given that we are stuck with this additional variability, can we use it in a useful manner, or if not, how can we make sure it does little harm?

Acceptance kernel $\pi(D|X)$ as an extension of modelling

Using ABC is equivalent to adding additional variability into the model.

- $\exists$ many interesting papers saying how to make this variability small
- Instead ask, given that we are stuck with this additional variability, can we use it in a useful manner, or if not, how can we make sure it does little harm?

How do we relate the simulator to the observations $\pi(D|S)$

- Measurement/sampling error on D
  - Measurement error may be built into the simulator. Could we remove it and use the ABC to do this?

Acceptance kernel $\pi(D|X)$ as an extension of modelling

Using ABC is equivalent to adding additional variability into the model.

- $\exists$ many interesting papers saying how to make this variability small
- Instead ask, given that we are stuck with this additional variability, can we use it in a useful manner, or if not, how can we make sure it does little harm?

How do we relate the simulator to the observations $\pi(D|S)$

- Measurement/sampling error on D
  - Measurement error may be built into the simulator. Could we remove it and use the ABC to do this?
- Discrepancy between the simulator and reality
  - In a deterministic model setting, Goldstein and Rougier 2008, and Kennedy and O'Hagan 2001 (amongst others), have offered advice for thinking about model discrepancies.
  - For statistical models, simulator error is a less clear concept.

Acceptance kernel $\pi(D|X)$ as an extension of modelling

Using ABC is equivalent to adding additional variability into the model.

- $\exists$ many interesting papers saying how to make this variability small
- Instead ask, given that we are stuck with this additional variability, can we use it in a useful manner, or if not, how can we make sure it does little harm?

How do we relate the simulator to the observations $\pi(D|S)$

- Measurement/sampling error on D
  - Measurement error may be built into the simulator. Could we remove it and use the ABC to do this?
- Discrepancy between the simulator and reality
  - In a deterministic model setting, Goldstein and Rougier 2008, and Kennedy and O'Hagan 2001 (amongst others), have offered advice for thinking about model discrepancies.
  - For statistical models, simulator error is a less clear concept.

The other useful way to view $\pi(D|X)$ is as a way of smoothing the likelihood function to make inference tractable.

Wood (2010)

Simon Wood introduced an ABC-like algorithm, for doing inference in models which have unknown likelihood.
The key idea is to introduce a synthetic Gaussian likelihood function for the simulator, and then use MCMC to find the posterior.

Wood 2010

Suppose our MCMC chain is currently at θ_i.

- Propose a move to θ' from some kernel
- Run the simulator n times at θ', giving realisations $X_1, \ldots, X_n$
- Summarize these to get summaries $S_1, \ldots, S_n$.
- Assume that $s \sim N(\mu_{\theta'}, \Sigma_{\theta'})$, and estimate $\mu_{\theta'}$ and $\Sigma_{\theta'}$.
- Assign θ' likelihood $\phi(s^{obs}; \mu_{\theta'}, \Sigma_{\theta'})$ and accept or reject θ' according the MH acceptance ratio.

Relationship between ABC and Simon Wood's approach

One way to view Wood 2010 is as an ABC algorithm, but using μ_θ and Σ_θ as the summary of $f(\theta)$, and assuming

$$\pi(D|S) = \exp(-\frac{1}{2}(D-\mu_\theta)^T\Sigma_\theta^{-1}(D-\mu_\theta))$$

An IS-GABC algorithm version of Wood 2010 is

- Pick $\theta \sim \pi(\theta)$
- Simulate $s_1, \ldots, s_n \sim f(\theta)$, calculate μ_θ and Σ_θ.
- Give θ weight $w = \pi(D|S)$

Relationship between ABC and Simon Wood's approach

One way to view Wood 2010 is as an ABC algorithm, but using μ_θ and Σ_θ as the summary of $f(\theta)$, and assuming

$$\pi(D|S) = \exp(-\frac{1}{2}(D-\mu_\theta)^T\Sigma_\theta^{-1}(D-\mu_\theta))$$

An IS-GABC algorithm version of Wood 2010 is

- Pick $\theta \sim \pi(\theta)$
- Simulate $s_1, \ldots, s_n \sim f(\theta)$, calculate μ_θ and Σ_θ.
- Give θ weight $w = \pi(D|S)$

This can be seen as accounting for the variability of the model run repeatedly at the same input, and then assuming the distribution is Gaussian.

Alternatively view: smoothing the simulator likelihood make inference more tractable.

Problems with Monte Carlo methods

Monte Carlo methods are generally guaranteed to succeed if we run them for long enough.

This guarantee comes at a cost.

- Most methods sample naively - they don't learn from previous simulations.
- They don't exploit known properties of the likelihood function, such as continuity
- They sample randomly, rather than using space filling designs.

This naivety can make a full analysis infeasible without access to a large amount of computational resource.

Problems with Monte Carlo methods

Monte Carlo methods are generally guaranteed to succeed if we run them for long enough.

This guarantee comes at a cost.

- Most methods sample naively - they don't learn from previous simulations.
- They don't exploit known properties of the likelihood function, such as continuity
- They sample randomly, rather than using space filling designs.

This naivety can make a full analysis infeasible without access to a large amount of computational resource.

If we are prepared to lose the guarantee of eventual success, we can exploit the continuity of the likelihood function to learn about its shape, and to dramatically improve the efficiency of our computations.

Likelihood estimation

The GABC framework assumes

$$\begin{aligned}\pi(D|\theta) &= \int \pi(D|X)\pi(X|\theta)\mathrm{dX} \\ &\approx \frac{1}{N}\sum \pi(D|X_i)\end{aligned}$$

where $X_i \sim \pi(X|\theta)$. Or in Wood (2010),

$$\pi(D|\theta) = \phi(D; \mu_\theta, \Sigma_\theta)$$

Likelihood estimation

The GABC framework assumes

$$\begin{aligned}\pi(D|\theta) &= \int \pi(D|X)\pi(X|\theta)\mathrm{dX} \\ &\approx \frac{1}{N}\sum \pi(D|X_i)\end{aligned}$$

where $X_i \sim \pi(X|\theta)$. Or in Wood (2010),

$$\pi(D|\theta) = \phi(D; \mu_\theta, \Sigma_\theta)$$

For many problems, we believe the likelihood is continuous and smooth, so that $\pi(D|\theta)$ is similar to $\pi(D|\theta')$ when $\theta - \theta'$ is small

We can model $\pi(D|\theta)$ and use the model to find the posterior in place of running the simulator.

Example: Ricker Model

The Ricker model is one of the prototypic ecological models.

- used to model the fluctuation of the observed number of animals in some population over time
- It has complex dynamics and likelihood, despite its simple mathematical form.

Ricker Model

- Let N_t denote the number of animals at time t.

$$N_{t+1} = rN_t e^{-N_t + e_r}$$

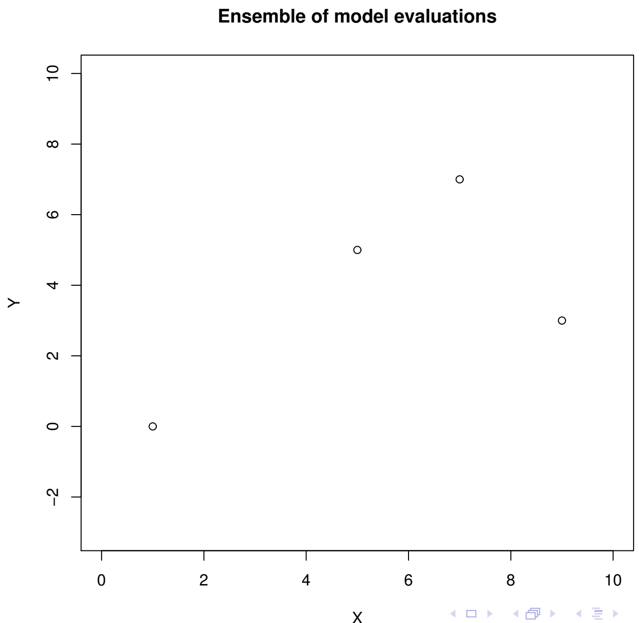
where e_t are independent $N(0, \sigma_e^2)$ process noise

- Assume we observe counts y_t where

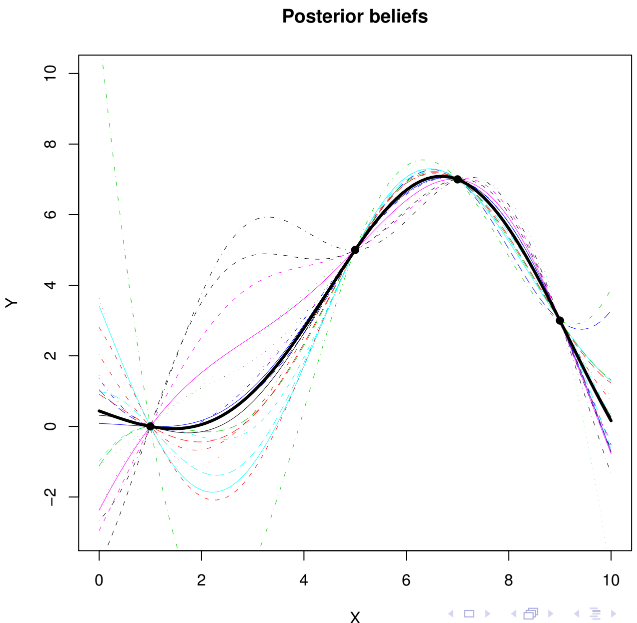
$$y_t \sim Po(\phi N_t)$$

Used in Wood to demonstrate the synthetic likelihood approach.

Gaussian Process Illustration



Gaussian Process Illustration



GP emulator choices

$$\eta(x) = h(x)\beta + u(x)$$

$$\text{emulator} = \text{mean structure} + \text{residual}$$

GP emulator choices

$$\eta(x) = h(x)\beta + u(x)$$
$$\text{emulator} = \text{mean structure} + \text{residual}$$

We take $u(x)$ to be a zero-mean Gaussian process

$$u(\cdot) \sim GP(0, c(\cdot,\cdot))$$

with squared exponential covariance function with a nugget term.

GP emulator choices

$$\eta(x) = h(x)\beta + u(x)$$
$$\text{emulator} = \text{mean structure} + \text{residual}$$

We take $u(x)$ to be a zero-mean Gaussian process

$$u(\cdot) \sim GP(0, c(\cdot, \cdot))$$

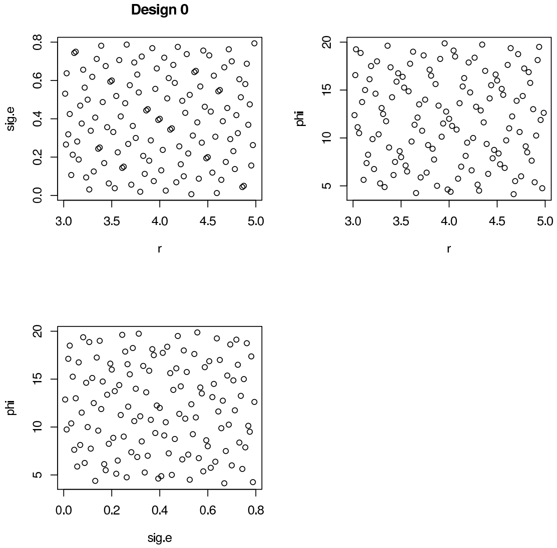
with squared exponential covariance function with a nugget term.

We let the mean function $h(x)$ include up to quadratic polynomial terms, as typically we know

$$\log(\pi(D|\theta)) \to -\infty \text{ as } \theta \to \pm\infty$$

Design 1 - 128 pts

We use a Sobol sequence on the prior input space to find a design $\{\theta_i\}_{i=1}^{d}$. We estimate the likelihood at each point in the design, and aim to fit a GP model to estimate the likelihood at θ values not in the design.



Difficulties

i. The likelihood is too difficult to model, so we model the log-likelihood instead.

$$\hat{l}(D|\theta) = \log\left(\frac{1}{N}\sum \pi(D|X_i)\right)$$

ii. Use bootstrapped replicates of the log-likelihood to estimate the variance of the nugget term (we could estimate it as part of the GP fitting, but typically this is very poorly behaved).

Difficulties

i. The likelihood is too difficult to model, so we model the log-likelihood instead.

$$\hat{l}(D|\theta) = \log\left(\frac{1}{N}\sum \pi(D|X_i)\right)$$

ii. Use bootstrapped replicates of the log-likelihood to estimate the variance of the nugget term (we could estimate it as part of the GP fitting, but typically this is very poorly behaved).

iii. $\mathbb{V}\mathrm{ar}(\hat{l}(D|\theta))$ is far from constant as a function of θ - this causes a problem as simple GP covariance functions assume the nugget is constant through space.

   1. Crude fix: pick a small sensible value estimated for a θ value near the mode of the posterior.

History matching waves

The log-likelihood for a typical problem ranges across too wide a range of values, e.g., -10 near the mode, but essentially $-\infty$ at the extremes of the prior range.

Consequently, any Gaussian process model will struggle to model the log-likelihood across the entire input range.

History matching waves

The log-likelihood for a typical problem ranges across too wide a range of values, e.g., -10 near the mode, but essentially $-\infty$ at the extremes of the prior range.
Consequently, any Gaussian process model will struggle to model the log-likelihood across the entire input range.

- To fix this we introduce the idea of waves, similar to those used in Michael Goldstein's approach to history-matching.
- In each wave, we build a GP model that can rule out large swathes of space as *implausible*.

History matching waves

The log-likelihood for a typical problem ranges across too wide a range of values, e.g., -10 near the mode, but essentially $-\infty$ at the extremes of the prior range.
Consequently, any Gaussian process model will struggle to model the log-likelihood across the entire input range.

- To fix this we introduce the idea of waves, similar to those used in Michael Goldstein's approach to history-matching.
- In each wave, we build a GP model that can rule out large swathes of space as *implausible*.
  We decide that θ is implausible if

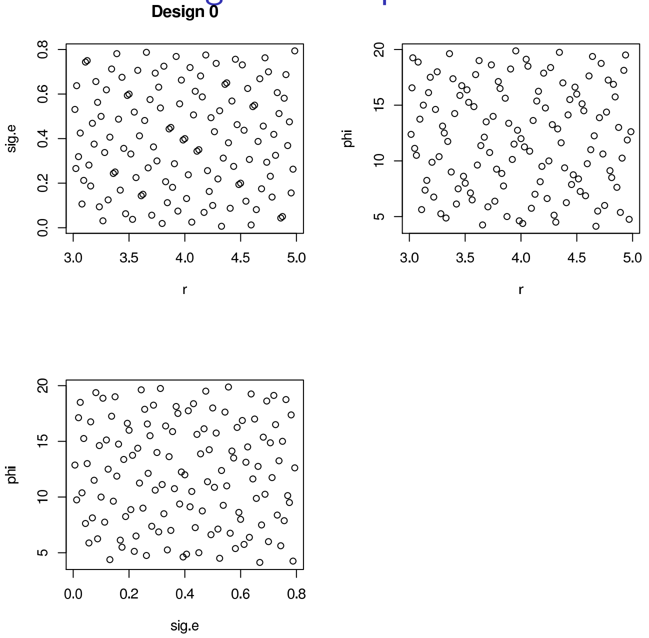
  $$m(\theta) + 3\sigma < \max_{\theta_i} \log \pi(D|\theta_i) - 10$$

  where $m(\theta)$ is the Gaussian process estimate of $\log \pi(D|\theta)$, and σ is the variance of the GP estimate.
  - We subtract 10, as for the Ricker model, a difference of 10 on the log scale between two likelihoods, means that assigning the θ with the smaller log-likelihood a posterior density of 0 (by saying it is implausible) is a good approximation.

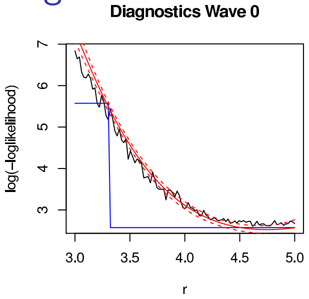
Difficulties II ctd

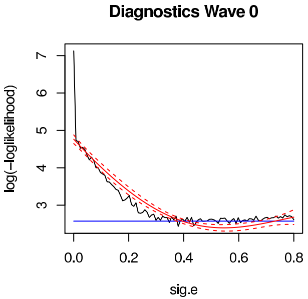
- This still wasn't enough in some problems, so for the first wave we model $\log(-\log \pi(D|\theta))$
- For the next wave, we begin by using the Gaussian processes from the previous waves to decide which parts of the input space are implausible.
- We then extend the design into the not-implaussible range and build a new Gaussian process
- This new GP will lead to a new definition of implausibility
- ...

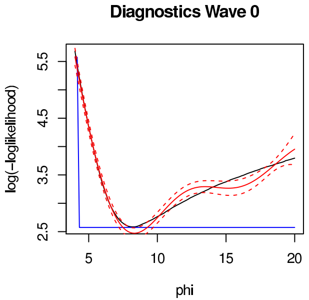
Results - Design 1 - 128 pts



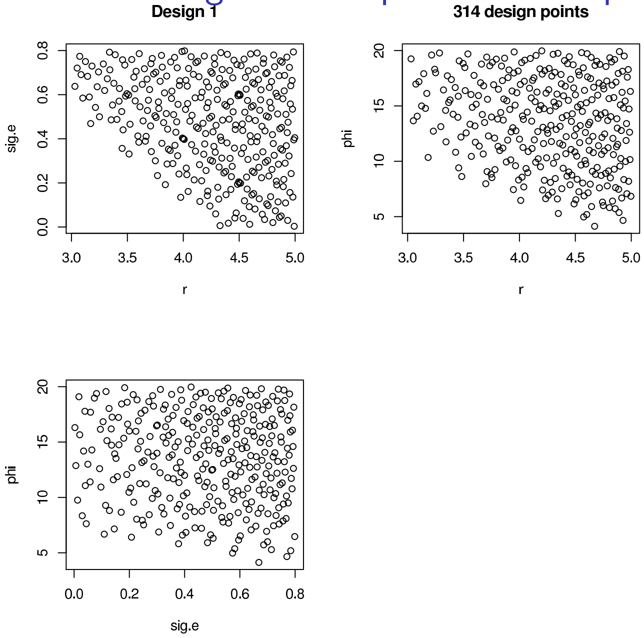
Diagnostics for GP 1 - threshold = 5.6



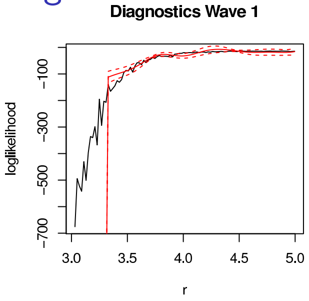


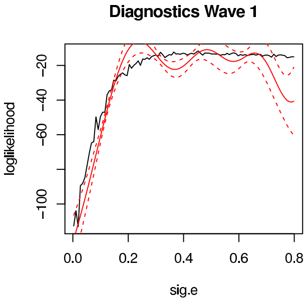


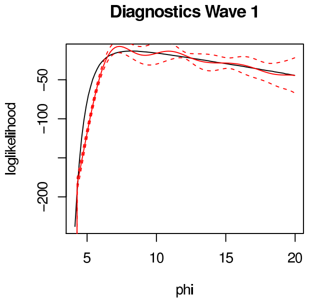
Results - Design 2 - 314 pts - 38% of space implausible



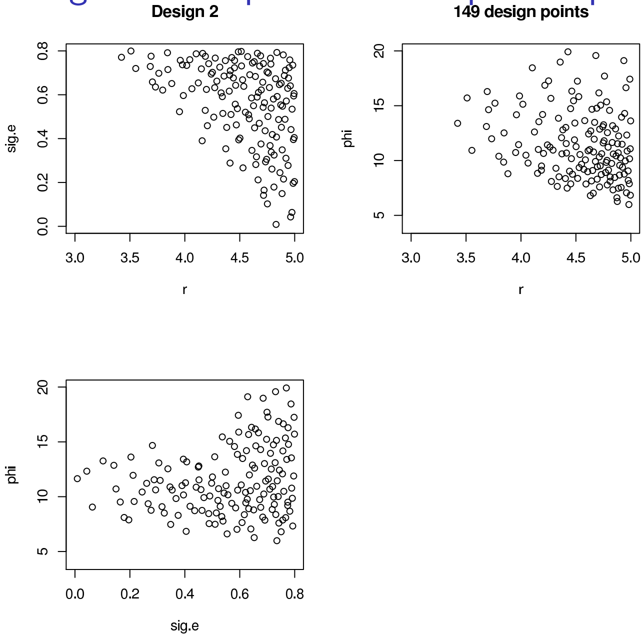
Diagnostics for GP 2 - threshold = -21.8



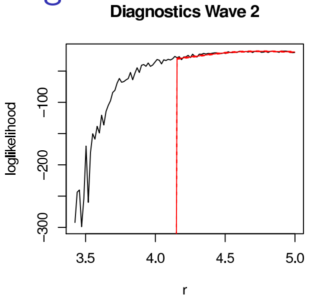


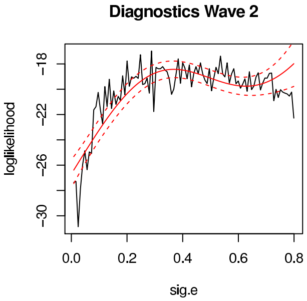


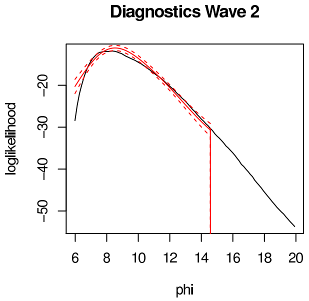
Design 3 - 149 pts - 62% of space implausible



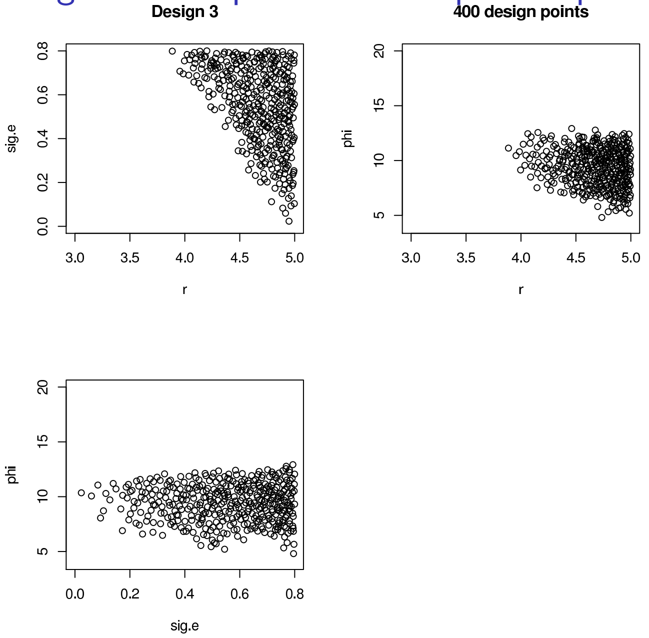
Diagnostics for GP 3 - threshold = -20.7



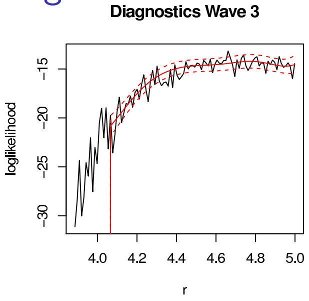


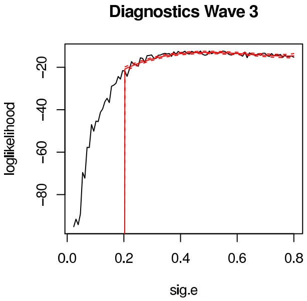


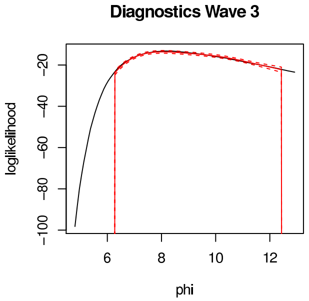
Design 4 - 400 pts - 95% of space implausible



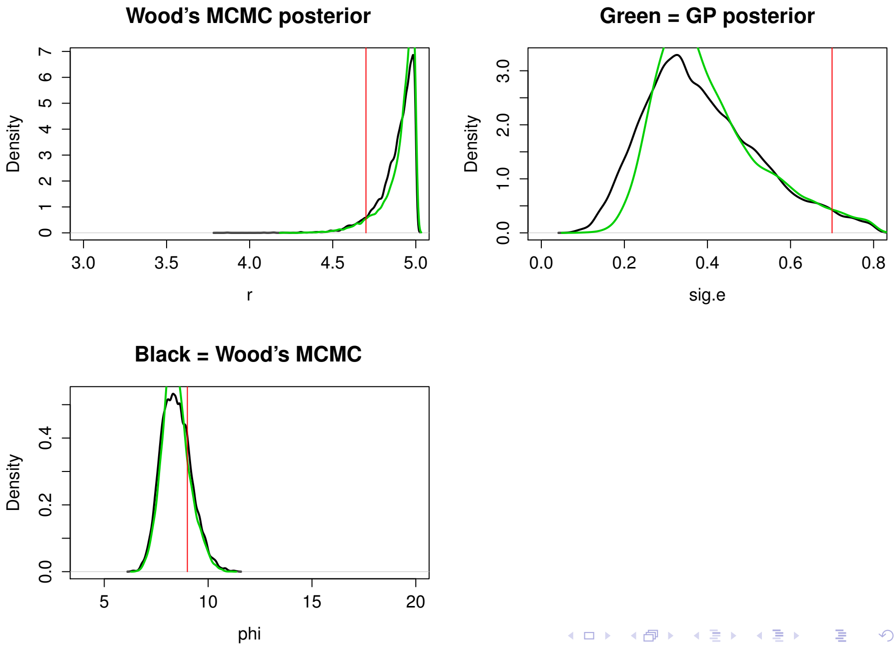
Diagnostics for GP 4 - threshold = -16.4







MCMC Results



Computational details

- The Wood MCMC method used $10^5 \times 500$ simulator runs
- The GP code used $(128 + 314 + 149 + 400) = 991 \times 500$ simulator runs
    - 1/100th of the number used by Wood's method.

By the final iteration, the Gaussian processes had ruled out over 98% of the original input space as implausible,

- the MCMC sampler did not need to waste time exploring those regions.

Computational details

- The Wood MCMC method used $10^5 \times 500$ simulator runs
- The GP code used $(128 + 314 + 149 + 400) = 991 \times 500$ simulator runs
  - 1/100th of the number used by Wood's method.

By the final iteration, the Gaussian processes had ruled out over 98% of the original input space as implausible,

- the MCMC sampler did not need to waste time exploring those regions.

Unfortunately though, GPs are computationally expensive to train.

The CPU time taken to run both methods was approximately the same!

For more complex models, there will hopefully be time advantages.

Conclusions

- Monte Carlo methods are naive
  - they don't learn
  - they don't exploit continuity or design considerations

  This makes them powerful, as they will always give the correct answer in time.
- However, computational resource is usually limited.
- If we believe the likelihood is a continuous function of the parameters, and we're prepared to sacrifice asymptotic perfection in the hope of achieving a good approximation in finite time, then we can use Gaussian processes to accelerate the inference process.
- Lots still to do
  - Nugget issues
  - diagnostic checking
  - justification of threshold values
  - ...

Conclusions

- Monte Carlo methods are naive
  - they don't learn
  - they don't exploit continuity or design considerations

  This makes them powerful, as they will always give the correct answer in time.
- However, computational resource is usually limited.
- If we believe the likelihood is a continuous function of the parameters, and we're prepared to sacrifice asymptotic perfection in the hope of achieving a good approximation in finite time, then we can use Gaussian processes to accelerate the inference process.
- Lots still to do
  - Nugget issues
  - diagnostic checking
  - justification of threshold values
  - ...

Thank you for listening!