Scalable algorithms for Markov process parameter inference

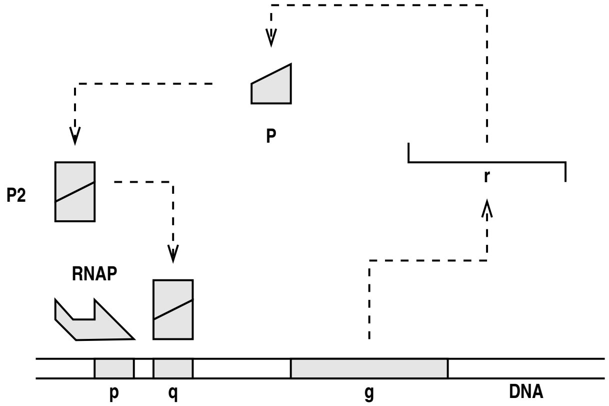
Darren Wilkinson
@darrenjw
tinyurl.com/darrenjw
Newcastle University, U.K.
and
The Alan Turing Institute, U.K.

Connecting models and data in the life sciences
BIRS, Banff, Alberta, Canada
12th November 2018

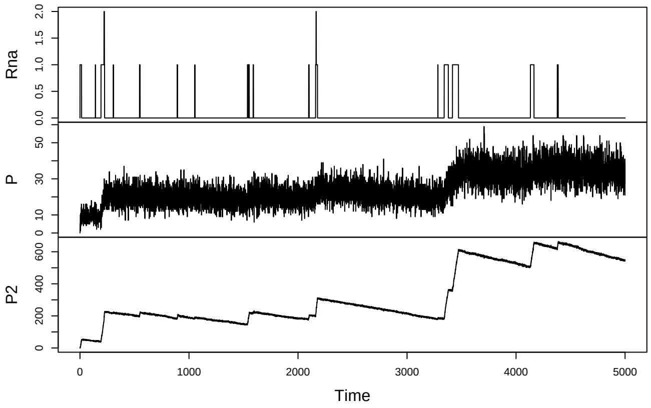
Overview

- Stochastic reaction networks, stochastic simulation and partially observed Markov process (POMP) models
- Modularity: separating model representation from simulation algorithm
- Well-mixed versus reaction–diffusion
- Likelihood-free MCMC for POMP models: separating simulation from inference
- Likelihood-free PMMH pMCMC, ABC and ABC–SMC
- Functional programming approaches for scalable scientific and statistical computing

Example — genetic auto-regulation



Simulated realisation of the auto-regulatory network



Modularity: decoupling models from simulation algorithms

- For forward modelling, there are clear and considerable benefits to separating the representation of the model from the algorithm used to simulate realisations from it:
    - There are numerous exact and approximate simulation algorithms, as well as algorithms for static model analysis — by decoupling the models from the algorithms, it is easy to apply any algorithm to any model of interest — improvements in simulation algorithms automatically apply to all models of interest
    - Modifying a model representation will typically be much easier than modifying an algorithm to simulate that model
    - When assembling large models from smaller model components, it is often more straightforward to compose model representations than associated simulation algorithms
- Few disadvantages: limit to flexibility of representation, slight inefficiencies, more sophisticated programming required

Lotka–Volterra system

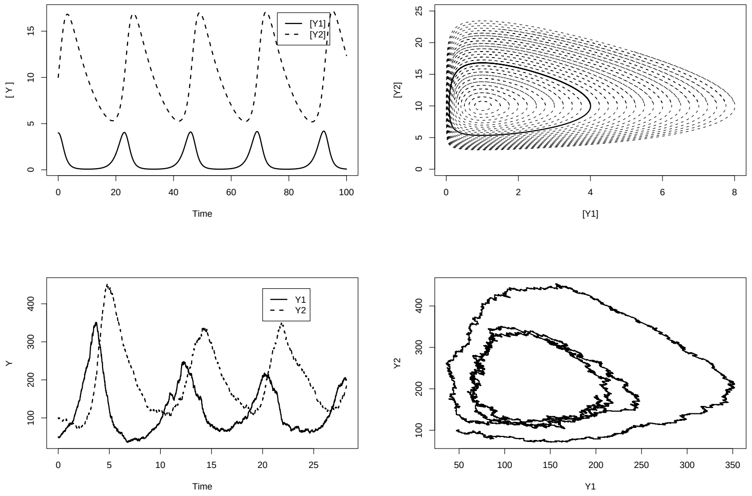
Trivial (familiar) example from population dynamics, but here the "reactions" are elementary biochemical reactions taking place inside a cell

Reactions

$$\begin{aligned} X &\longrightarrow 2X && \text{(prey reproduction)} \\ X+Y &\longrightarrow 2Y && \text{(prey-predator interaction)} \\ Y &\longrightarrow \emptyset && \text{(predator death)} \end{aligned}$$

- X – Prey, Y – Predator
- We can re-write this using matrix notation

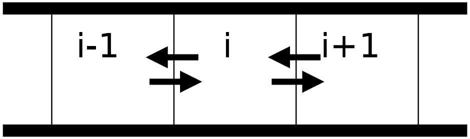
The Lotka-Volterra model



The well-mixed assumption

- The fundamental assumption underpinning the mass-action stochastic kinetic approach to modelling chemical reactions as a Markov jump process is that the hazard associated with any reaction event is constant
- It is this assumption of constant hazard which leads to exponential inter-arrival times of reaction events and all of the other algorithms commonly used for non-spatial modelling
- However, it's pretty clear that molecules far apart will have a lower reaction hazard than molecules which are nearby
- Mass-action kinetics assumes that molecular diffusion is rapid relative to the time scales associated with the chemical reactions
- Evidence that this assumption is violated for many interesting intra-cellular processes

Stochastic kinetics of diffusion

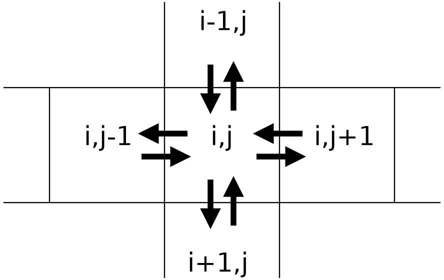


- We can think of diffusion events as "reactions":

$$\mathcal{X}_i \longrightarrow \mathcal{X}_{i+1}$$
$$\mathcal{X}_i \longrightarrow \mathcal{X}_{i-1}$$

- There are 2 reactions per sub-volume, so $2N$ reactions in total (for periodic boundary conditions)
- This defines a Markov jump process which we can solve exactly or simulate using the Gillespie algorithm

Discrete stochastic diffusion on a 2d lattice

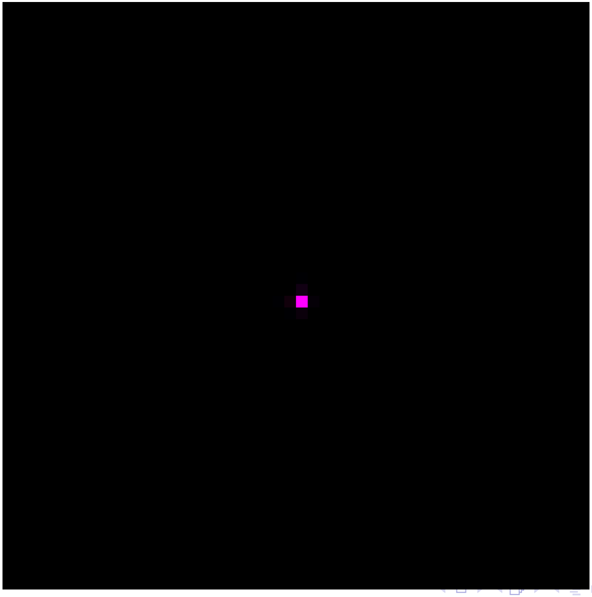


$$B_1 X_{i,j,t} \equiv X_{i-1,j,t}, \quad B_2 X_{i,j,t} \equiv X_{i,j-1,t}$$

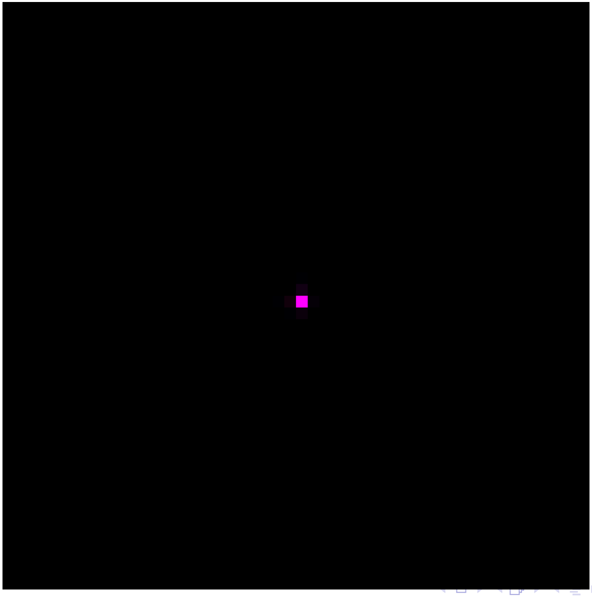
$$\nabla_i \equiv 1 - B_i, \quad \Delta \equiv \nabla_1^2 B_1^{-1} + \nabla_2^2 B_2^{-1}$$

So $\Delta X_{i,j,t} = X_{i-1,j,t} + X_{i+1,j,t} + X_{i,j-1,t} + X_{i,j+1,t} - 4X_{i,j,t}$

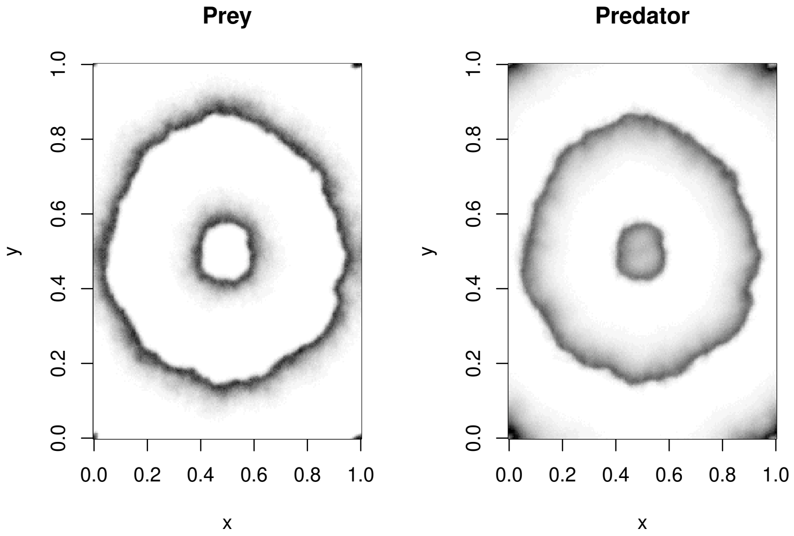
Discrete stochastic reaction diffusion on a 2d lattice



Lotka–Volterra SPDE dynamics (via the spatial CLE)



Lotka–Volterra reaction–diffusion SPDE



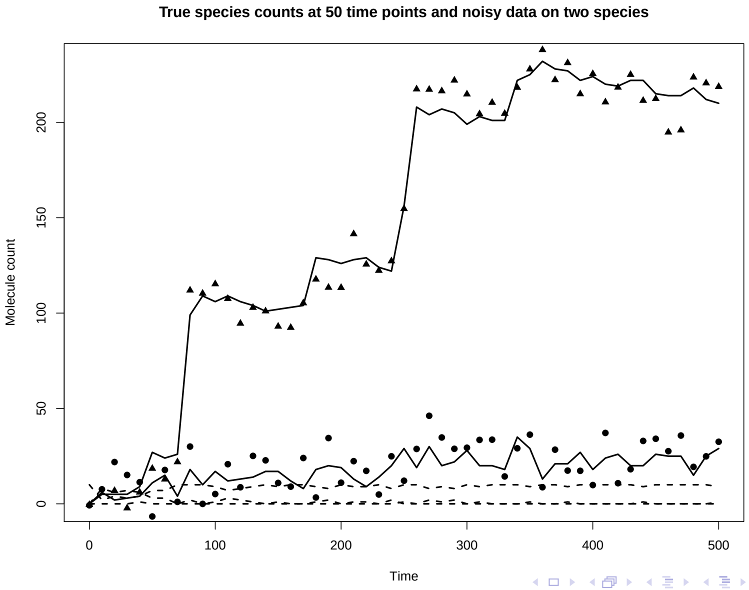
Modularity and simulation algorithms

- Separating models from simulation algorithms has many benefits
- Separating model parameters from models so that simulation algorithms don't need to know about parameters is similarly beneficial
- Models can be simulated in different ways, using different algorithms, and under different assumptions: exact/approximate, discrete/continuous, stochastic/deterministic, well-mixed/spatial, ...
- Model exchange formats, such as SBML, can be useful for understanding some of the issues

Parameter inference

- The auto-regulatory network model contains 5 species and 8 reactions
- Each reaction has an associated rate constant — these 8 rate constants may be subject to uncertainty
- The initial state of the model (5 species levels) may also be uncertain/unknown
- There could also be uncertainty about the structure of the reaction network itself — eg. presence/absence of particular reactions — this can be embedded into the parameter inference problem, but is often considered separately, and is not the subject of this talk
- We will focus here on using time course data on some aspect of one (or more) realisations of the underlying stochastic process in order to make inferences for any unknown parameters of the model

Partial, noisy data on the auto-reg model



Classes of Bayesian Monte Carlo algorithms

In this context there are 3 main classes of MC algorithms:

- ABC algorithms (likelihood–free)
  - Completely general (in principle) “global” Approximate Bayesian Computation algorithms, so just require a forward simulator, and don’t rely on (eg.) Markov property, but typically very inefficient and approximate
- POMP algorithms (likelihood–free)
  - Typically “local” (particle) MCMC-based algorithms for Partially Observed Markov Processes, again only requiring a forward simulator, but using the Markov property of the process for improved computational efficiency and “exactness”
- Likelihood-based MCMC algorithms
  - More efficient (exact) MCMC algorithms for POMP models, working directly with the model representation, not using a forward simulator, and requiring the evaluation of likelihoods associated with the sample paths of the stochastic process

Modularity and model decoupling for inference

- Decoupling the model from the inference algorithm is just as important as separation of the model from a forward simulation algorithm
- The key characteristic of likelihood-free (or "plug-and-play") algorithms is that they separate inference algorithms from the forward simulator completely — this strong decoupling has many advantages, with the main disadvantage being the relative inefficiency of the inferential algorithms
- The likelihood-free algorithms rely heavily on forward simulation, so can immediately benefit from improvements in exact and approximate simulation technology
- There is no reason why efficient likelihood-based MCMC algorithms can't also be decoupled from the model representation, but doing so for a reasonably large and flexible class of models seems to be beyond the programming skills of most statisticians...

Partially observed Markov process (POMP) models

- Continuous-time Markov process: $\mathbf{X} = \{X_s | s \geq 0\}$ (for now, we suppress dependence on parameters, θ)
- Think about integer time observations (extension to arbitrary times is trivial): for $t \in \mathbb{N}$, $\quad \mathbf{X}_t = \{X_s | t-1 < s \leq t\}$
- Sample-path likelihoods such as $\pi(\mathbf{x}_t | x_{t-1})$ can often (but not always) be computed (but are often computationally difficult), but discrete time transitions such as $\pi(x_t | x_{t-1})$ are typically intractable
- Partial observations: $\mathcal{Y} = \{y_t | t = 1, 2, \ldots, T\}$ where

$$y_t | X_t = x_t \sim \pi(y_t | x_t), \qquad t = 1, \ldots, T,$$

where we assume that $\pi(y_t | x_t)$ can be evaluated directly (simple measurement error model)

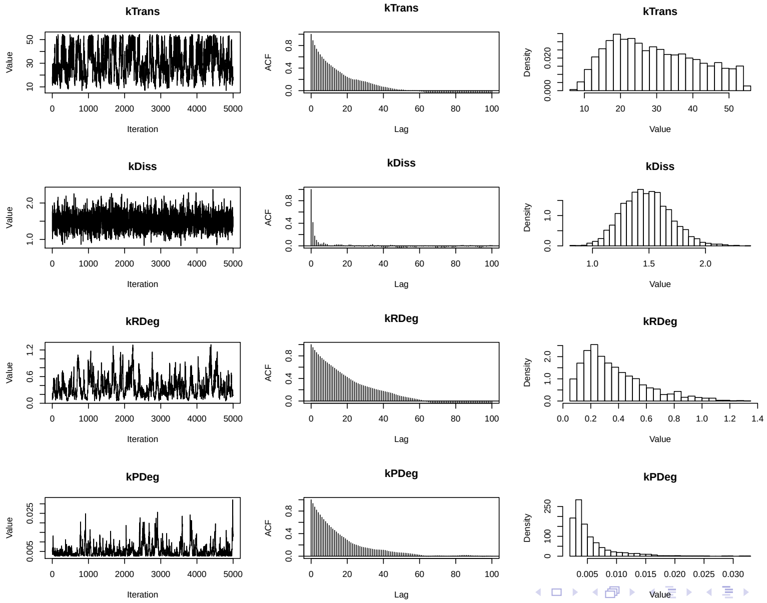
Bayesian inference for latent process models

- Vector of model parameters, θ, the object of inference
- Prior probability distribution on θ, denoted $\pi(\theta)$
- Conditional on θ, we can simulate realisation of the stochastic process $\mathbf{X}$, with probability model $\pi(\mathbf{x}|\theta)$, which may be intractable
- Observational data $\mathcal{Y}$, determined from $\mathbf{x}$ and θ by a the probability model $\pi(\mathcal{Y}|\mathbf{x},\theta)$ — for "exact" algorithms we typically require that this model is tractable, but for ABC, we only need to be able to simulate from it
- Joint model $\pi(\theta,\mathbf{x},\mathcal{Y}) = \pi(\theta)\pi(\mathbf{x}|\theta)\pi(\mathcal{Y}|\mathbf{x},\theta)$
- Posterior distribution $\pi(\theta,\mathbf{x}|\mathcal{Y}) \propto \pi(\theta,\mathbf{x},\mathcal{Y})$
- If using Monte Carlo methods, easy to marginalise out $\mathbf{x}$ from samples from the posterior to get samples from the parameter posterior $\pi(\theta|\mathcal{Y})$

Likelihood-free PMMH pMCMC

- Particle Markov chain Monte Carlo (pMCMC) methods are a powerful tool for parameter inference in POMP models
- In the variant known as particle marginal Metropolis Hastings (PMMH), a (random walk) MH MCMC algorithm is used to explore parameter space, but at each iteration, a (bootstrap) particle filter (SMC algorithm) is run to calculate terms required in the acceptance probability
- The "magic" of pMCMC is that despite the fact that the particle filters are "approximate", pMCMC algorithms nevertheless have the "exact" posterior distribution of interest (either $\pi(\theta|\mathcal{Y})$ or $\pi(\theta, \mathbf{x}|\mathcal{Y})$) as their target
- If a sophisticated particle filter is used, pMCMC can be a reasonably efficient likelihood-based MCMC method — however, when a simple "bootstrap" particle filter is used, the entire process is "likelihood-free", but still "exact"

PMMH inference results



“Sticking” and tuning of PMMH

- As well as tuning the θ proposal variance, it is necessary to tune the number of particles, N in the particle filter — need enough to prevent the chain from sticking, but computational cost roughly linear in N
- Number of particles necessary depends on θ, but don't know θ *a priori*
- Initialising the sampler is non-trivial, since much of parameter space is likely to lead to likelihood estimates which are dominated by noise — how to move around when you don't know which way is “up”?!
- Without careful tuning and initialisation, burn-in, convergence and mixing can all be very problematic, making algorithms painfully slow...

Alternative: approximate Bayesian computation (ABC)

- Since $\pi(\theta, \mathbf{x}, \mathcal{Y}) = \pi(\theta)\pi(\mathbf{x}|\theta)\pi(\mathcal{Y}|\theta, \mathbf{x})$, it is trivial to generate samples from $\pi(\theta, \mathbf{x}, \mathcal{Y})$ and to marginalise these down to $\pi(\theta, \mathcal{Y})$
- Exact rejection algorithm: generate $(\theta^\star, \mathcal{Y}^\star)$ from $\pi(\theta, \mathcal{Y})$ and keep provided that $\mathcal{Y} = \mathcal{Y}^\star$ otherwise reject and try again
- This gives exact realisations from $\pi(\theta|\mathcal{Y})$, but in practice the acceptance rate will be very small (or zero)
- ABC: Define a metric on the sample space, $\rho(\cdot,\cdot)$, and accept $(\theta^\star, \mathcal{Y}^\star)$ if $\rho(\mathcal{Y}, \mathcal{Y}^\star) < \varepsilon$
- This gives exact realisations from $\pi(\theta|\rho(\mathcal{Y}, \mathcal{Y}^\star) < \varepsilon)$, which tends to the true posterior as $\varepsilon \longrightarrow 0$
- Still problematic if there is a large discrepancy between the prior and posterior...

Summary statistics

- The choice of metric $\rho(\cdot, \cdot)$ is very important to the overall efficiency and performance of ABC methods
- Using a naive Euclidean distance on the raw data $\rho(\mathcal{Y}, \mathcal{Y}^\star) = \|\mathcal{Y} - \mathcal{Y}^\star\|$ is likely to perform poorly in practice — even with a perfect choice of parameters, it is extremely unlikely that you will "hit" the data
- Ideally, we would use a vector of sufficient statistics, $s(\mathcal{Y})$ of the likelihood model associated with the process, to summarise the important aspects of the data relevant to parameter inference, and then define a metric on $s(\cdot)$
- In practice, for complex models we don't know the sufficient statistics (and they probably don't exist), but we nevertheless form a vector of summary statistics, which we hope capture the important aspects of the process and ignore the irrelevant noise in each realisation

Issues with simple rejection ABC

- There are two main problems with naive rejection sampling based ABC:
  - The first relates to the dimension of the data, and this is (largely) dealt with by carefully choosing and weighting appropriate summary statistics
  - The second relates to the dimension of the parameter space...
- If the dimension of the parameter space is large, the posterior distribution is likely to have almost all of its mass concentrated in a tiny part of the space covered by the prior, so the chances of hitting on good parameters when sampling from the prior will be very small
- Might be better to gradually "zoom in" on promising parts of the parameter space gradually over a series of iterations...

ABC–SMC

- Interest in a Bayesian posterior distribution

$$\pi(\theta|x) \propto \pi(\theta)f(x|\theta)$$

  where $f(x|\theta)$ is intractable
- Observed data x_0
- Sequence of approximations

$$\pi_t(\theta) = \pi(\theta|\rho(x, x_0) < \varepsilon_t),$$

  where $\infty = \varepsilon_0 > \varepsilon_1 > \cdots > \varepsilon_n > 0$ and $\rho(\cdot, \cdot)$ is a suitable metric on data space
- π_0 is the prior, and for sufficiently small ε_n, hopefully π_n not too far from the posterior, $\pi(\theta|x_0)$
- Progressively reduce tolerances to improve agreement between successive distributions and hopefully improve acceptance rates

Pros and cons of ABC(–SMC)

- All likelihood-free methods have a tendency to be very computationally intensive and somewhat inefficient
- ABC is very general, and can be applied to arbitrary settings (eg. not just POMP models)
- ABC methods parallelise very well, and hence can be useful for getting reasonably good approximations to be true posterior relatively quickly if suitable hardware is available
- ABC is becoming increasingly popular outside of statistics, where the idea of "moment matching" is familiar and intuitive
- ABC usually results in a distribution significantly over-dispersed relative to the true posterior
- The tuning parameters can affect the ABC posterior
- It's hard to know how well you are doing when working "blind"

Pros and cons of pMCMC

- Most obvious application is to POMP models — less general than ABC
- It targets the “exact” posterior distribution, irrespective of the choices of tuning constants!
- In practice, for finite length runs, the pMCMC output tends to be slightly under-dispersed relative to the true posterior (“missing the tails”)
- Parallelises fine over multiple cores on a single machine, but less well over a cluster
- Although the theory underpinning pMCMC is non-trivial, implementing likelihood-free PMMH is straightforward, and has the advantage that it targets the “exact” posterior distribution

Likelihood free inference

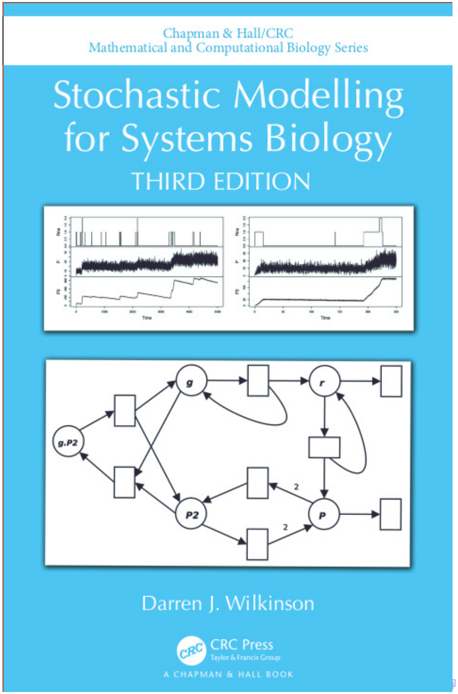
- For conducting Bayesian inference for complex simulation models, “likelihood–free” methods are very attractive
- There are many likelihood–free algorithms, some of which are “exact” — pMCMC algorithms being a notable example
- Likelihood-free algorithms can sometimes be very inefficient
- pMCMC is not the only option worth considering — ABC–SMC methods, and SMC^2 are also worth trying; also iterated filtering for a ML solution
- The reliance of likelihood free algorithms on forward simulation fundamentally limits their effectiveness and utility for many challenging problems — inference is fundamentally about conditional simulation — other ways of modularising models and inferential algorithms are also worth considering

Composable models and algorithms

- We want to construct models and algorithms from composable and inter-changable pieces
- Pure, referentially transparent (mathematical) functions are exactly the right abstraction for this purpose
- Functional programming languages encourage and support the use of pure functions for constructing programs
- Functions encourage the separation of concerns:
  - although we often parametrise models, a function for constructing a simulation algorithm from a fully specified model shouldn't know if or how that model is parametrised
  - Similarly, a function for running a bootstrap particle filter for a simulation model, shouldn't need to know anything about the model structure, let alone if or how it is parametrised
  - A PMMH algorithm should just accept a noisy likelihood function, and shouldn't know anything about how it is constructed

Nesting and composing algorithms

- Problems associated with parameter inference and model selection typically involve composing many layers of algorithm together
- For a typical LF-PMMH algorithm:
  - A parameter vector is drawn from a prior distribution
  - The parameter vector is used to fully-specify a Markov process model
  - The Markov process model representation is used to generate a transition kernel which can be forward simulated using an appropriate algorithm
  - The transition kernel is embedded into a bootstap particle filter for marginal likelihood estimation
  - The marginal likelihood evaluator is embedded into a Metropolis-Hastings algorithm
- Most people working in this field aren't trained in how to write flexible generic software for solving these kinds of multi-layered problems



Chapman & Hall/CRC
Mathematical and Computational Biology Series

Stochastic Modelling for Systems Biology

THIRD EDITION

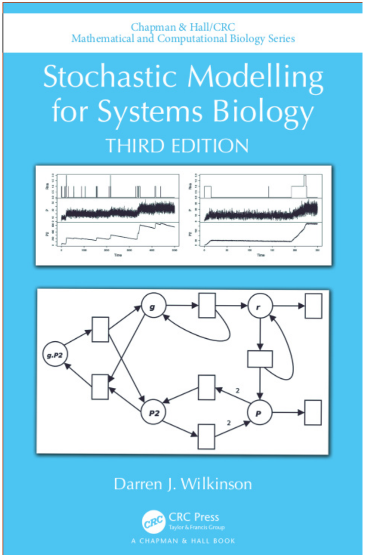
Darren J. Wilkinson

CRC Press
Taylor & Francis Group

A CHAPMAN & HALL BOOK

SMfSB3e: New in the third edition

- New chapter on spatially extended systems, covering the spatial Gillespie algorithm for reaction diffusion master equation (RDME) models in 1- and 2-d, the next subvolume method, spatial CLE, scaling issues, etc.
- Significantly expanded chapter on inference for stochastic kinetic models from data, covering approximate methods of inference (ABC), including ABC-SMC. The material relating to particle MCMC has also been improved and extended.
- Updated R package, including code relating to all of the new material
- New R package for parsing SBML models into simulatable stochastic Petri net models
- New software library, written in Scala, replicating most of the functionality of the R packages in a fast, compiled, strongly typed, functional language


Chapman & Hall/CRC
Mathematical and Computational Biology Series
Stochastic Modelling for Systems Biology
THIRD EDITION
Darren J. Wilkinson
CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

- In the press right now
- Published 29th November
- Pre-order now for Xmas!
- New website/GitHub repository
- Lots of new, improved and updated software — all free, open source, well-documented and available now

`https://github.com/darrenjw/smfsb`