Computational Statistics (2021) 36:2863–2887
https://doi.org/10.1007/s00180-021-01102-6

ORIGINAL PAPER

Direct statistical inference for finite Markov jump processes via the matrix exponential

Chris Sherlock[1]

Received: 20 July 2020 / Accepted: 23 March 2021 / Published online: 19 April 2021

Abstract
Given noisy, partial observations of a time-homogeneous, finite-statespace Markov chain, conceptually simple, direct statistical inference is available, in theory, via its rate matrix, or infinitesimal generator, Q, since $\exp(Qt)$ is the transition matrix over time t. However, perhaps because of inadequate tools for matrix exponentiation in programming languages commonly used amongst statisticians or a belief that the necessary calculations are prohibitively expensive, statistical inference for continuous-time Markov chains with a large but finite state space is typically conducted via particle MCMC or other relatively complex inference schemes. When, as in many applications Q arises from a reaction network, it is usually sparse. We describe variations on known algorithms which allow fast, robust and accurate evaluation of the product of a non-negative vector with the exponential of a large, sparse rate matrix. Our implementation uses relatively recently developed, efficient, linear algebra tools that take advantage of such sparsity. We demonstrate the straightforward statistical application of the key algorithm on a model for the mixing of two alleles in a population and on the Susceptible-Infectious-Removed epidemic model.

Keywords Markov jump process · Likelihood inference · Bayesian inference · Matrix exponential

1 Introduction

A *reaction network* is a stochastic model for the joint evolution of one or more populations of *species*. These species may be chemical or biological species (e.g. Wilkinson 2012), animal species (e.g. Drovandi and McCutchan 2016), interacting groups of individuals at various stages of a disease (e.g.Andersson and Britton 2000), or counts of sub-populations of alleles (e.g. Moran 1958), for example. The state of the system

✉ Chris Sherlock
c.sherlock@lancaster.ac.uk

[1] Department of Mathematics and Statistics, Lancaster University, Lancaster, UK

is encapsulated by the number of each species that is present, and the system evolves via a set of *reactions*: Poisson processes whose rates depend on the current state.

Typically, partial and/or noisy observations of the state are available at a set of time points, and statistical interest lies in inference on the unknown rate parameters, the filtering estimate of the state of the system after the latest observation or prediction of the future evolution of the system. The usual method of choice for exact inference on discretely observed Markov jump processes (MJPs) on a finite or countably infinite state space is Bayesian inference via particle Markov chain Monte Carlo (particle MCMC, Andrieu et al. (2010)) using a bootstrap particle filter (e.g. Andrieu et al. (2009); Golightly and Wilkinson (2011); Wilkinson (2012); McKinley et al. (2014); Owen et al. (2015); Koblents and Miguez (2015)). Other MCMC and SMC-based techniques are available e.g. Kypraios et al. (2017), and, a further latent-variable-based MCMC method when the statespace is finite Rao and Teh (2013).

Particle MCMC and SMC, however, are relatively complex algorithms, even more so when a bootstrap particle filter (simulation from the process itself) is not suitable and a bridge simulator is necessary, such as when observation noise is small or when there is considerable variability in the state from one observation to the next; see Golightly and Wilkinson (2015), Golightly and Sherlock (2019), Black (2019). In cases where the number of states, d, is finite, direct exact likelihood-based inference is available via the exponential of the infinitesimal generator for the continuous-time Markov chain, or rate matrix, Q. Whilst such inference is conceptually straightforward, it has often been avoided in practice for general MJPs, except in cases where the number of states is very small e.g. Amoros et al. (2019). The computational cost of each iteration of particle MCMC is proportional to the number of particles used and, for efficient estimation; see Doucet et al. (2015),Sherlock et al. (2015) this is approximately linear in the size of the statespace, d. In contrast, Matrix exponentiation has a computational cost of $\mathcal{O}(d^3)$, which, together with a lack of suitable tools in `R`, could explain the lack of uptake of this method. However, conceptually simple statistical inference via the matrix exponential is entirely practical in many cases even when the number of states is in the thousands or higher, and it has been used successfully in a subclass of these situations (e.g. Jenkinson and Goutsias (2012), see Sect. 2.2). There are three main reasons why this is possible:

1. Matrix exponentials themselves are never needed; only the *product* of a vector and a matrix exponential is ever required.
2. The matrices to be exponentiated are infinitesimal generators and, as such, have a *special structure*; furthermore, the vector that pre-multiplies the matrix exponential is non-negative.
3. The matrices to be exponentiated are usually *sparse*; tools for basic operations with large, sparse matrices in `C++` and interfacing the resulting code with `R` have recently become widely available; see Eddelbuettel and Sanderson (2014), Sanderson and Curtin (2018).

The sparsity of Q arises because the number of possible 'next' states given the current state is bounded by the number of reactions, which is typically small. This article describes matrix exponential algorithms suitable for statistical application in many cases, and demonstrates their use for inference, filtering and prediction. Associ-

ated code provides easy-to-use R interfaces to C++ implementations of the algorithms, which are typically simpler and often faster than more generally applicable algorithms for matrix exponentiation.

Section 1.1 describes the Susceptible-Infectious-Removed (SIR) model for the evolution of an infectious disease and the Moran model for the mixing of two alleles in a population, then briefly mentions many more such models where the statespace is finite, and a few where it is countably infinite. The two main examples will be used to benchmark and illustrate the techniques in this article. As well as being directly of use for models with finite state spaces, exponentials of finite rate matrices can also be used to perform inference on Markov jump processes with a countably infinite statespace; see Georgoulas et al. (2017) and Sherlock and Golightly (2019). The latter uses the uniformisation and scaling and squaring algorithms as described in this article, while the former uses the less efficient but more general algorithm of Al-Mohy and Higham (2011) (see Sect. 3).

Section 2 of this article presents the likelihood for discretely and partially observed data on a finite-statespace continuous-time Markov chain and presents two 'tricks' specific to epidemic models, that allow for a massive reduction in the size of the generators that are needed compared with the size of the statespace. Section 3 describes the Matrix exponential algorithms and Sect. 4 benchmarks some of the algorithms and demonstrates their use for inference, filtering and prediction. The article concludes in Sect. 5 with a discussion.

1.1 Examples and motivation

Both by way of motivation and because we shall use them later to illustrate our method, we now present two examples of continuous-time Markov processes, where a finite, sparse rate matrix contains all of the information about the dynamics.

For each Markov process, the set of possible states can be placed in one-to-one correspondance with a subset of the non-negative integers $\{1, \ldots, d\}$. The off-diagonal elements of the rate matrix, Q, are all non-negative, and the ith diagonal element is $\mathsf{Q}_{ii} = -\sum_{j=1, j\neq i}^{d} \mathsf{Q}_{i,j}$. A chain that is currently in state i leaves this state upon the first event of a Poisson process with a rate of $-\mathsf{Q}_{i,i}$; the state to which it transitions is j with a probability of $\mathsf{Q}_{i,j}/(-\mathsf{Q}_{i,i})$. Whilst the rate matrix, Q, is a natural description of the process, the likelihood for typical observation regimes involves the transition matrix, $\exp(\mathsf{Q}t)$, the (i, j)th element of which is exactly $\mathbb{P}(X_t = j | X_0 = i)$.

Both examples take the form of a *reaction network*, where from the current state X_t, the next state change will occur according to one of the specified reactions. The state can be thought of as an integer vector, where each element of the vector indicates the numbers of a particular species that are currently present in the system. When, as here, the maximum number of each species is finite, the set of possible states can be placed in one-to-one correspondence with the natural numbers as required to define Q. Each reaction occurs according to a Poisson process with a rate, $\lambda(X_t)$, and when it occurs species combine according to the reaction formula. For example, the first reaction in the SIR model, below occurs with a rate of βSI, and when it occurs the state (S, I) changes to $(S-1, I+1)$.

Example 1 **The SIR model for epidemics**. The SIR model for a disease epidemic has 3 species: those who are susceptible to the epidemic, S, those both infected and infectious, I, and those who have recovered from the epidemic and play no further part in the dynamics, R. The non-negative counts of each species are denoted by S, I, and R. For relatively short epidemics the population, n_{pop}, is assumed to be fixed, and so the state of the Markov chain, represented by (S, I), is subject to the additional constraint of $S + I \leq n_{pop}$, with $R = n_{pop} - S - I$. The two possible reactions and their associated rates are:

$$\mathsf{S} + \mathsf{I} \xrightarrow{\beta SI} 2\mathsf{I}, \quad \text{and} \quad \mathsf{I} \xrightarrow{\gamma I} \mathsf{R}.$$

Example 2 **The Moran model for allele frequency** descibes the time evolution of the frequency of two alleles, A_1 and A_2 in a population with a fixed size of n_{pop}. Individuals with allele A_1 reproduce at a rate of α, and those with A_2 reproduce at a rate of β. When an individual dies it is replaced by the offspring of a parent chosen uniformly at random from the whole population (including the individual that dies). The allele that the parent passes to the offspring usually matches its own, however as it is passed down an allele may mutate; allele A_1 switching to A_2 with a probability of u and A_2 switching to A_1 with a probability of v. Let A_1 and A_2 represent individuals with alleles A_1 and A_2 respectively and let N be the number of individuals with allele A_1. The two reactions are

$$\mathsf{A}_1 \xrightarrow{\lambda_N} \mathsf{A}_2 \quad \text{and} \quad \mathsf{A}_2 \xrightarrow{\mu_N} \mathsf{A}_1.$$

Setting $f_N = N/n_{pop}$, the corresponding infinitesimal rates are

$$\begin{aligned} \lambda_N &= (1 - f_N)\left[\alpha f_N(1 - u) + \beta(1 - f_N)v\right] \quad \text{and} \\ \mu_N &= f_N\left[\beta(1 - f_N)(1 - v) + \alpha f_N u\right], \end{aligned}$$

where the unit of time is the expectation of the exponentially distributed time for an individual to die and be replaced. □

The many other examples of interest include the SIS and SEIR models for epidemics (e.g. Andersson and Britton 2000), dimerisation and the Michaelis-Menten reaction kinetics (e.g. Wilkinson 2012). Further examples but with an infinite statespace include the Schlögel model (e.g. Vellela and Qian 2009), the Lotka-Volterra predator-prey model (e.g. Wilkinson 2012, Drovandi and McCutchan 2016) and models for the autoregulation of the production of a protein (e.g. Wilkinson 2012), all of which are tackled using matrix exponentials in Sherlock and Golightly (2019).

2 Data and likelihood calculations

Denote the statespace of the Markov chain $\{X_t\}_{t\geq 0}$ by $\mathcal{X} = \{x^{(k)}\}_{k=1}^{d}$. Let the prior mass function across states be $\nu(x|\theta)$ and define $\nu(\theta) := (\nu(x^{(1)}|\theta), \ldots, \nu(x^{(d)}|\theta))$.

Let the infinitesimal generator be $\mathsf{Q}(\theta)$, and suppose there are observations $y_0, y_1, \ldots, y_n$ at times $t_0, t_1, \ldots, t_n$, where $Y_i|(X_i = x_i)$ has a mass function of $p(y_i|x_i, \theta)$, $i = 0, \ldots, n$.

2.1 Likelihood for noisy and partially observed data

For any continuous-time Markov chain $\{X_t\}_{t\geq 0}$ with an infinitesimal generator, or rate matrix of Q, the $(x, x')th$ element of $\exp(\mathsf{Q}t)$ gives the transition probability (e.g. Norris (1997)):

$$\mathbb{P}\left(X_t = x' \mid X_0 = x\right) = \left[\exp(\mathsf{Q}t)\right]_{x,x'},$$

where here and elsewhere we abuse notation by identifying the state $x^{(i)} \in \mathcal{X}$ with the corresponding index $i \in \{1, \ldots, d\}$.

Defining the diagonal likelihood matrix to be $L_j(\theta) = \mathsf{diag}(p(y_j|x^{(1)}, \theta), \ldots, p(y_j|x^{(d)}, \theta))$ and $\Delta_j = t_j - t_{j-1}$, $j = 1, \ldots, n$, the likelihood for the observations is then

$$\begin{aligned}\mathbb{P}(y_0, \ldots, y_n \mid \theta) = &\sum_{(x_0,\ldots,x_n)\in\mathcal{X}^{n+1}} \mathbb{P}(X_0 = x_0)\,\mathbb{P}(Y_0 = y_0|X_0 = x_0) \\ &\prod_{j=1}^{n} \mathbb{P}\left(X_j = x_j|X_{j-1} = x_{j-1}\right)\mathbb{P}\left(Y_j = y_j|X_j = x_j\right) \\ &= \nu(\theta)^\top L_0(\theta)\left[\prod_{j=1}^{n} \exp(\mathsf{Q}(\theta)\Delta_j)L_j(\theta)\right]\underline{1}, \end{aligned} \tag{1}$$

where $\underline{1}$ is the d-vector of ones. Similarly, the filtering distribution after observation y_m is

$$\mathbb{P}\left(X_{t_m} = x \mid y_0, \ldots, y_m\right) = \frac{\nu(\theta)^\top L_0(\theta)\left[\prod_{j=1}^{m} \exp(\mathsf{Q}(\theta)\Delta_j)L_j(\theta)\right]}{\nu(\theta)^\top L_0(\theta)\left[\prod_{j=1}^{m} \exp(\mathsf{Q}(\theta)\Delta_j)L_j(\theta)\right]\underline{1}}. \tag{2}$$

Consider the required multiplication from left to right: since the likelihood vectors $L_j(\theta)$ are diagonal, pre-multiplication by a d-vector is an $\mathcal{O}(d)$ operation. Pre-multiplication of the exponential of a sparse matrix by a d-vector via the uniformisation algorithm is also $\mathcal{O}(d)$ (see Sect. 3.1), so the entire likelihood calculation is $\mathcal{O}(d)$. In the case of certain epidemic models d itself can be much smaller than might naively be assumed.

2.2 Statespace reduction for epidemic models

An alternative formulation of the statespace of the Markov chain for an SIR epidemic model (or more general models such as the SEIR), in terms of the *degree of*

advancement (DA), was first pointed out in Jenkinson and Goutsias (2012). Instead of representing the state in terms of the number of susceptibles and the number of infecteds, given a known initial condition it is represented by the number of new infections and the number of new removals, B_I and B_R, neither of which can be negative and both of which are non-decreasing. Given the initial condition, the map from (S, I) to (B_I, B_R) is one-to-one; however the rate matrix with the DA formulation is lower triangular, a key ingredient in the implicit Euler integration scheme used in Jenkinson and Goutsias (2012) to integrate the master equation.

When performing Bayesian inference for the SIR model using noisy, partial observations, Ho et al. (2018) points out that augmenting the state space of the MCMC Markov chain to include not just the model parameters but also the true values of S and I at each observation time can massively reduce the sizes of the statespaces that need to be considered when evolving the SIR process from one observation time to the next provided the DA formulation is used. Consider the case of exact observations and suppose, for example, that in a population of size $n_{pop} = 500$, $x_a = (S_a, I_a, R_a) = (485, 2, 13)$ and for some $t > 0$, $x_{a+t} = (470, 3, 27)$. Then $b_R = R_{a+t} - R_a = 14$ and $b_I = S_a - S_{a+t} = 15$. The size of the statespace for evolution between time a and time $a+t$, $\mathcal{X}_a^{a+t}$, is then reduced from the size of the full statespace, $(n_{pop}+1)(n_{pop}+2)/2 = 125751$ to $(b_I+1)(b_R+1) = 240$. The exponential of the rate matrix is not used in Ho et al. (2018); instead, a recursive formula for the Laplace transform of the transition probability to a given new state in terms of transition probabilities for old states then permits estimation of the transition vector from a known initial starting point in $\mathcal{O}(d)$ operations, where d is the dimension of the statespace actually required. Inference is then performed for the SIR model using data from the Ebola outbreak in regions of Guinea.

We may use the DA formulation with data augmentation, provided we include an additional coffin state, C, with $Q_{\mathsf{C},x} = 0$ for all $x \in \mathcal{X}_a^{a+t} \cup \mathsf{C}$. Any births that would leave the statespace (and hence contradict the observation at time $a + t$) instead go to C. The aforementioned implementation, a square grid of possible states, includes "impossible" states to which the rate of entry is zero: the current number of infections can never be negative, so, throughout the time interval $[a, a + t]$, $b_R \leq I_a + b_I$. Removing these states altogether allows us to make a further reduction in the size of the statespace, by a factor of up to one half. In the example above, this reduces the statespace size still further, from 240 to 162.

3 Matrix exponentiation

The exponential of a $d \times d$ square matrix, M is defined via its infinite series: $e^{\mathsf{M}} = \sum_{i=0}^{\infty} \frac{1}{i!}\mathsf{M}^i$. As might be anticipated from the definition, for a $d \times d$ matrix, algorithms for evaluating $\exp(\mathsf{M})$ take $\mathcal{O}(d^3)$ operations (see Moler and Van Loan (2003), for a review of many such methods). However, for a d-vector, v, the product $\exp(Mt)\, v$ is the solution to the initial value problem $w(0) = v$, $dw/dt = \mathsf{M}w$, and is the key component of the solution to more complex differential equations such as $dw/dt = \mathsf{M}w + Bu(t)$. For this reason the numerical evaluation of the action of a

matrix exponential on a vector has received considerable attention of itself, e.g. Gallopoulos and Saad (1992), Saad (1992), Sidje (1998), Al-Mohy and Higham (2011).

When M is dense,

$$\exp(\mathsf{M})\, v = \sum_{i=0}^{\infty} \frac{1}{i!} \mathsf{M}^i v \tag{3}$$

can be evaluated in $\mathcal{O}(d^2)$ operations if the series is truncated at an appropriate point. However, motivated by the examples in Sect. 1.1 our interest lies in large *sparse* matrices, and the number of operations can then be reduced to $\mathcal{O}(rd)$, where r is the average number of entries in each row of M.

With double-precision arithmetic, real numbers are stored to an accuracy of approximately 10^{-16}. Thus, evaluation of the exponential of a large negative number via its Taylor series is prone to potentially enormous round-off errors due to the almost cancellation of successive large positive and negative terms; a similar problem can affect the exponentiation of a matrix. Such issues are typically circumvented via the identity

$$\exp(\mathsf{M})v = \left[\prod_{k=1}^{K} \exp(\mathsf{M}/K) \right] v, \tag{4}$$

applied for a sufficiently large integer K, and evaluated via K successive evaluations of product of $\exp(\mathsf{M}/k)$ and a vector. The calculation on the right of (4) typically involves many more numerical operations than the direct calculation on the right of (3), so K should be the *smallest* integer that leads to the required precision by mitigating sufficiently against the cancellation of large positive and negative terms. This minimises both the accumulation of rounding errors and the total compute time given the required accuracy.

One common technique for such multiplication, exemplified in the popular `Expokit FORTRAN` routines of Sidje (1998), estimates $e^{\mathsf{M}/K} v$ via its projection on to the Krylov subspace of $\mathsf{Span}\{v, \mathsf{M}v, \ldots, \mathsf{M}^{n-1}v\}$, where $n << d$. A second method is provided in Al-Mohy and Higham (2011), where the key contributions lie in the method for choosing K and for choosing a suitable truncation point for the infinite series, as well as a means of truncating each series early depending on the behaviour of recent terms. These and other algorithms are compared, specifically for the case of the SIR model (which has a special structure; see Sect. 2.2) in Kinyanjui et al. (2018).

Both Krylov methods and that of Al-Mohy and Higham (2011) use the fact that M is sparse and that only the action of $\exp(\mathsf{M})$ on a vector is required, but neither uses the structure of interest to us: we require $v^\top \exp(\mathsf{Q}t)$ where Q is a rate matrix for a general MJP and v is a non-negative vector. Since $\mathsf{Q}t$ is also a rate matrix, we henceforth set $t = 1$ without loss of generality. Let

$$\rho := \max_{i=1,\ldots,d} |\mathsf{Q}_{ii}| \quad \text{and} \quad \mathsf{P} = (1/\rho)\mathsf{Q} + I. \tag{5}$$

P is a Markov transition matrix, and the key observation is that

$$\exp \mathsf{Q} = \exp(\rho \mathsf{P} - \rho \mathsf{I}) = \exp(-\rho)\exp(\rho \mathsf{P}) = \sum_{i=0}^{\infty} \exp(-\rho)\frac{\rho^i}{i!}\mathsf{P}^i. \tag{6}$$

Firstly, P has no negative entries so cancellation of terms with alternating signs is no longer a concern. Secondly, $\exp \mathsf{Q}$ can be interpreted as a mixture over a Poisson(ρ) random variable I, of I transitions of the discrete-time Markov chain with a transition matrix of P.

The next two subsections detail variations on two existing algorithms that utilise this special structure: the *uniformisation* algorithm and a variation on the *scaling and squaring* algorithm. For sparse rate matrices, the uniformisation algorithm has a cost of $\mathcal{O}(\rho d)$, whereas the scaling and squaring algorithm has a cost of $\mathcal{O}(d^3 \log \rho)$. Thus, the uniformisation algorithm is preferred when ρ is small, and scaling and squaring when ρ is large but d is relatively small. We now describe the two algorithms in detail.

3.1 The uniformisation algorithm

In many statistical applications, the most appropriate algorithm for calculating $\mu^\top := \nu^\top \exp \mathsf{Q}$ is the uniformisation algorithm, e.g. Reibman and Trivedi (1988), Sidje and Stewart (1999). This estimates $\mu^\top$ by truncating a single series none of whose terms can be negative, rather than truncating multiple series where terms may change sign as in Al-Mohy and Higham (2011). Given an $\epsilon > 0$, the algorithm calculates an approximation, $\widehat{\mu}$, to μ by picking a truncation point for the infinite series, such that, if ν were a probability vector, the (guaranteed to be non-negative) amount of *true* missing probability over all of the d dimensions is controlled:

$$0 < 1 - \frac{||\widehat{\mu}^*||_1}{||\nu||_1} < \epsilon,$$

where $\widehat{\mu}^*$ is the probability vector that would be calculated if there were no rounding errors, and the only errors were due to the truncation of the infinite series. Typically we aim for ϵ to be similar to the machine's precision. We control the absolute truncation error and note that with any truncation of the power series, it is impossible to obtain general control of the *relative error in a given component* of μ, $|\widehat{\mu}_i/\mu_i - 1|$. Consider, for example, a Moran process (Example 2), where Q is tridiagonal. Then Q^k is also banded, with a band width of $2k+1$. For any given m_{max}, and $\nu = (1, 0, 0, \ldots)$, set $d > m_{max} + 1$. The truncated approximation to e^{Q} gives a transition probability of 0 for all states above $m_{max} + 1$, yet, in truth there is a non-zero probability of such a transition. However, the combined probability of all transitions which have not been accounted for is, by design, at most ϵ.

From (6),

$$\mu^\top = \nu^\top e^{Q} = e^{-\rho}\nu^\top \sum_{i=0}^{\infty} \frac{\rho^i}{i!} P^i \approx e^{-\rho} \sum_{i=0}^{m} \frac{\rho^i}{i!} \nu^\top P^i =: \widehat{\mu}^{*\top}.$$

Now,

$$\sum_{i=1}^{d} \widehat{\mu}_i^* = \widehat{\mu}^{*\top} 1 = e^{-\rho} \sum_{i=0}^{m} \frac{\rho^i}{i!} \nu^\top P^i 1 = ||\nu||_1 e^{-\rho} \sum_{i=0}^{m} \frac{\rho^i}{i!}.$$

So the absolute relative error, or (when ν is a probability vector) missing probability mass, due to truncation is

$$r_m(\rho) := e^{-\rho} \sum_{i=m+1}^{\infty} \frac{\rho^i}{i!},$$

the tail probability of a $\mathsf{Poisson}(\rho)$ random variable. Of direct interest to us is

$$m_\epsilon(\rho) := \inf\{m \in \mathbb{N} : r_m(\rho) \leq \epsilon\},$$

the smallest m required to achieve an error of at most ϵ, or, essentially, the quantile function for a $\mathsf{Poisson}(\rho)$ random variable, evaluated at $1-\epsilon$. Chebyshev's inequality applied to X/ρ, where $X \sim \mathsf{Poisson}(\rho)$ gives $\mathbb{P}\left(|X/\rho - 1| \geq 1/\sqrt{\epsilon\rho}\right) \leq \epsilon$, implying the $m = \mathcal{O}(\rho)$ computational cost given earlier in this section.

In many programming languages, standard functions are available to evaluate $m_\epsilon(\rho)$. However, for example, in `R` we find

```
> rho=100; eps=1e-16
> qpois(eps,rho,lower.tail=FALSE)
[1] Inf
> ppois(193,rho,lower.tail=FALSE) # 193 is correct answer, not infinity
[1] 5.713551e-17
> eps=1e-15
> qpois(eps,rho,lower.tail=FALSE)
[1] 185
> ppois(185,rho,lower.tail=FALSE)
[1] 1.035777e-14
> ppois(189,rho,lower.tail=FALSE) # 189 is correct answer, not 185
[1] 8.017165e-16
```

i.e., an inability to calculate $m_\epsilon(\rho)$ correctly given the small ϵ values that we require; the underlying functions are also callable from `C++` and lead to the same error. In Appendix A we provide sharp bounds on $m_\epsilon(\rho)$, and this leads to an accurate methodology for its exact calculation, producing the same (correct) answers as the `C++ boost` library (which we have not been able to use with `Rcpp`) and up to twice as quickly.

The uniformisation algorithm is presented as Algorithm 3.1. For large values of ρ, although there is no problem with large positive and negative terms cancelling, it is possible that the partial sum $\sum_{i=0}^{k} \frac{\rho^i}{i!}$ might exceed the largest floating point number storable on the machine. We circumvent this problem by occasionally renormalising the vector partial sum when the most recent contribution is large, and compensating for this at the end; see lines 5, 12 and 14.

Algorithm 1 Uniformisation algorithm for $v^\top e^{\mathsf{Q}}$ with a missing mass of at most ϵ.

$\rho \leftarrow \max_{i=1}^{d} |Q_{i,i}|$; $\mathsf{M} \leftarrow \mathsf{Q} + \rho \mathsf{I}_d$; $BIG \leftarrow 10^{100}$.
Find $m_\epsilon(\rho)$.
$b \leftarrow ||v||_1$; $c \leftarrow 0$.
if $b > BIG$ **then**
  $v \leftarrow v/b$; $c \leftarrow c + \log b$; $b \leftarrow 1$.
$v_{pro} \leftarrow v_{sum} \leftarrow v$.
$f \leftarrow 1$.
for j from 1 to m **do**
  $v_{pro}^\top \leftarrow v_{pro}^\top \mathsf{M}/f$; $b \leftarrow b\rho/f$.
  $v_{sum} \leftarrow v_{sum} + v_{pro}$.
  if $b > BIG$ **then**
    $v_{pro} \leftarrow v_{pro}/b$; $v_{sum} \leftarrow v_{sum}/b$; $c \leftarrow c + \log b$; $b \leftarrow 1$.
  $f \leftarrow f + 1$.
return $e^{c-\rho} \times v_{sum}$.

3.2 Scaling and squaring

One of the simplest, yet most robust methods for exponentiating any square matrix is the scaling and squaring algorithm, e.g. Moler and Van Loan (2003). When the square matrix is an infinitesimal generator, this method can be made even more robust using the reformulation in (6). Furthermore, when not $\exp \mathsf{Q}$ but $v^\top \exp \mathsf{Q}$ is required, some further computational savings can be obtained.

The basic scaling and squaring algorithm takes advantage of the identity

$$\exp(\mathsf{M}) = \left[\exp(\mathsf{M}/2^s)\right]^{2^s},$$

where for any integer s, a square matrix is raised to the power of 2^s by squaring it s times. We set $\mathsf{M} = \mathsf{Q} + \rho\mathsf{I} = \rho\mathsf{P}$ from (5). And define $\mathsf{M}_{small} = \mathsf{M}/2^s$. First, $\exp(\mathsf{M}_{small})$ is approximated via the uniformisation algorithm applied to a matrix, e.g. Ross (1996): $\sum_{i=0}^{m} \mathsf{M}_{small}^i/i!$. This quantity is then squared s times. The optimal value of s is chosen according to an algorithm described in Appendix B.

When evaluating $v^\top \exp(\mathsf{Q}) = \exp(-\rho)v^\top \exp(\mathsf{M})$ via scaling and squaring with $s > 0$ it is never most efficient to first evaluate $\exp(\mathsf{M})$. Let s_1 and s_2 be two integers such that $s_1 + s_2 = s$. Then

$$v^\top \exp(\mathsf{M}) = v^\top [\exp(\mathsf{M}_{small})]^{2^{s_1}} [\exp(\mathsf{M}_{small})]^{2^{s_1}} \ldots [\exp(\mathsf{M}_{small})]^{2^{s_1}},$$

with 2^{s_2} matrix vector products. The cost of s_1 matrix squares and 2^{s_2} vector-matrix products (where the matrix is dense) is $s_1 d^3 + 2^{s_2} d^2$. We round the minimiser down to the nearest integer for simplicity, setting

$$s_2 = \min\left(s, \lfloor (\log d - \log\log 2)/\log 2 \rfloor\right) \tag{7}$$

Even with $d = 2$ this gives $s_2 = \min(s, 1)$.

Algorithm 2 Scaling and squaring algorithm for $v^\top e^{\mathsf{Q}}$ with a missing mass of at most ϵ.

1: $\rho \leftarrow \max_{i=1}^{d} |Q_{i,i}|$.
2: Find s via linear search; $\rho_{small} \leftarrow \rho/2^s$; find $m_\epsilon(\rho_{small})$; find (s_1, s_2) via (7).
3: $\mathsf{M}_{small} \leftarrow (\mathsf{Q} + \rho\mathsf{I})/2^s$.
4: $v_{pro} \leftarrow v$.
5: $\mathsf{A}_{pro} \leftarrow \mathsf{M}_{small}$; $\mathsf{A}_{sum} \leftarrow \mathsf{I} + \mathsf{M}_{small}$
6: $f \leftarrow 2$.
7: **for** j from 2 to m **do**
8: $\quad \mathsf{A}_{pro} \leftarrow \mathsf{A}_{pro}\mathsf{M}_{small}/f$.
9: $\quad \mathsf{A}_{sum} \leftarrow \mathsf{A}_{sum} + \mathsf{A}_{pro}$.
10: $\quad f \leftarrow f + 1$.
11: $\mathsf{A}_{sum} \leftarrow e^{-\rho_{small}}\mathsf{A}_{sum}$
12: **for** j from 1 to s_1 **do**
13: $\quad \mathsf{A}_{sum} \leftarrow \mathsf{A}_{sum} \times \mathsf{A}_{sum}$.
14: **for** j from 1 to 2^{s_2} **do**
15: $\quad v_{pro}^\top \leftarrow v_{pro}^\top \mathsf{A}_{sum}$.
16: **return** $v_{pro}^\top$.

3.3 Improvements

We now describe two optional extensions: renormalisation, which improves the accuracy of any matrix exponentiation algorithm used on a rate matrix, and two-tailed truncation, which is unique to the uniformisation algorithm and allows a small computational saving.

Since $a := \sum_{i=1}^{d} \mu_i = \sum_{i=1}^{d} \nu_i$ there is no need to keep track of the logarithmic offset (c in Algorithm 3.1). Instead the final vector (v_{sum} in Algorithm 3.1) is renormalised at the end so that its components sum to a.

Two-tailed truncation, e.g. Reibman and Trivedi (1988) permits a small reduction in the computational cost of the uniformisation algorithm with no loss of accuracy. When ρ is moderate or large, the total mass of probability from the initial value of v_{sum} and the early values accumulated into v_{sum} (Steps 6 and 10 of Algorithm 3.1) is negligible (has a relative value smaller than $\epsilon/2$, say) compared with the sum of the later values. In such cases v_{sum} may be initialised to 0 and step 10 omitted for values of j beneath some m_{lo}. Proposition 1 below shows that if m is chosen such that $\mathbb{P}(\mathsf{Po}(\rho) > m) \leq \epsilon/2$ then setting $m_{lo} := \max(0, 2\lfloor \rho - 0.5 \rfloor - m)$ ensures that the missing probability mass is no more than ϵ. For large ρ, $m - m_{lo} = \mathcal{O}(\sqrt{\rho})$, so with two-tailed truncation the cumulative cost of Step 10 dwindles compared with the other $\mathcal{O}(d)$ costs, which are repeated $\mathcal{O}(\rho)$ times.

Proposition 1 *Given $\rho > 0$, let $p_n = e^{-\rho}\rho^n/n! = \mathbb{P}(\mathit{Poisson}(\rho)) = n$, and let $c = \lfloor \rho - 1/2 \rfloor$. Then for $a \leq c - 1$,*

$$\sum_{j=0}^{c-a-1} p_j < \sum_{j=c+a+1}^{\infty} p_j.$$

Proof For any integer b, and $1 \leq i \leq b$,

$$\begin{aligned}
&\frac{p_{b-i}}{p_{b+i}} \\
&= \rho^{-2i} b(b+1)(b-1)(b+2)\ldots(b-i+1)(b+i) \\
&= \rho^{-2i}\left[b_*^2 - \frac{1}{2^2}\right]\cdots\left[b_*^2 - \frac{(2i-1)^2}{2^2}\right]
\end{aligned}$$

where $b_* = b + 1/2$. Hence, if $b_* \leq \rho$, $p_{b-1}/p_{b+i} < 1$, and so

$$\sum_{j=0}^{\lfloor b^* \rfloor - a - 1} p_i = \sum_{i=a+1}^{\lfloor b_* \rfloor} p_{b-i} < \sum_{i=a+1}^{\lfloor b_* \rfloor} p_{b+i} < \sum_{i=a+1}^{\infty} p_{b+i}.$$

□

3.4 Implementation

Our `C++` implementation uses the recent basic sparse matrix functionality in the `C++ Armadillo` library; see Sanderson and Curtin (2016), Sanderson and Curtin (2018) to calculate $\nu^\top \exp \mathsf{Q}$, where ν is non-negative and Q is a large, sparse rate matrix. Direct function calls from the `R` programming language are enabled through `RcppArmadillo`; see Eddelbuettel and Sanderson (2014). For completeness, the functions can also be called with dense rate matrices. The functions are collected into the `expQ` package which is downloadable from https://github.com/ChrisGSherlock/expQ and are briefly outlined in Appendix C.

The speed of a vector multiplication by a sparse-matrix depends on the implementation of the sparse matrix algorithm. In `R`(R Core Team 2018) and in `C++ Armadillo`, sparse matrices are stored in column-major order. Hence pre-multiplication of the sparse matrix by a vector, $\nu^\top Q$, is much quicker than post multiplication, $Q\nu$. In other languages, such as `Matlab`, sparse matrices are stored in row-major order and post-multiplication is the quicker operation, so $Q^\top$ should be stored and used, rather than Q.

4 Numerical comparisons and demonstrations

In Al-Mohy and Higham (2011) their new algorithm (henceforth referred to as AMH) is compared across many examples against state-of-the-art competitors, including, in particular, the `expokit` function `expv` of Sidje (1998). In most of the experiments AMH is found to give comparable or superior accuracy together with superior computational speed. Given these existing comparisons and that the superiority of the uniformisation algorithm over the algorithm of Al-Mohy and Higham (2011) (for rate matrices) is not the main thrust of this paper, we perform a short comparison of accuracy and speed for two different likelihood calculations for an SIR model fitted

to data from the Eyam plague. We compare our implementation of the uniformisation algorithm, the algorithm of AMH, the `expAtv` function which is from the `R` package `expm` and uses the method of Sidje (1998), and the bespoke algorithm for epidemic processes in Ho et al. (2018). Since it would be unfair to compare the clock-speeds for the `Matlab` code for AMH directly with those of our `RcppArmadillo` implementation, we compare the number of sparse vector-matrix multiplications that are required.

When performing maximum-likelihood estimation, each iteration of the optimisation algorithm tries a new parameter value, and when performing Bayesian inference, each iteration of the algorithm proposes a new parameter value. In each case, given the parameter value, the pertinent rate matrices are created and then the matrix exponentiation function is called in turn for each of the matrices as required by (1). If the generic exponentiation function is called then the decision on whether to use Algorithm 1 or Algorithm 2 is based upon the dimension, d and the maximum absolute value on the diagonal, ρ. Whether Algorithm 1 or 2 is called directly or via the generic exponentiation function, the first task it performs is the evaluation of ρ. The cost of this is neglibible compared with that of the exponentiation itself, so it is essentially immaterial that ρ is evaluated twice when the generic exponentiation function is called.

The highest accuracy available in `C++` using sparse matrices and the `armadillo` linear algebra library is double precision, which we used throughout in our implementation of both of our algorithms. For the uniformisation and scaling and squaring algorithms we used $\epsilon = 10^{-15}$, and for AMH we used the double-precision option. For `expAtv` and for Ho et al. (2018) we use the default package setting.

4.1 Comparison with other matrix exponentiation algorithms

To examine the speed and accuracy of the algorithm we consider the collection (see the first three rows of Table 1) of (S, I) (susceptible and infected) values, which originated in Raggett (1982) and were used in Ho et al. (2018), for the Eyam plague. We set the parameters to their maximum-likelihood estimates, $(\beta, \gamma) = (0.0196, 3.204)$ and consider the likelihood for the data in Table 1. In addition, to mimic the size of potential changes between observation times and the size of the elements of the rate matrix from a larger population, we also evaluated the likelihood for the jump directly from the data at time 0 to the data at time 4. The final three rows of Table 1 refer to the rate matrix for the transition between consecutive observations and provide the dimension the matrix first using the reformulation of Ho et al. (2018) and then applying the improvement described in Sect. 2.2; the final row is the absolute value of the largest entry of Q, ρ. The rate matrix for the single jump between times 0 and 4 had $d_{HCS} = 30789$, $d = 16082$ and $\rho \approx 3439.5$. The full statespace has a size of 34453. Thus, for large changes, the main reduction in size arises from the improvement in Section 2.2, but for small jumps this provides a smaller relative reduction compared with that in Ho et al. (2018).

For the uniformisation and scaling and squaring algorithm, with $\epsilon = 10^{-15}$, the algorithm of Ho et al. (2018) and the `expAtv` function from the `R` package `expm` which uses the technique of Sidje (1998) we found the CPU time for 1000 estimations

Table 1 Time (in units of 31 days), and numbers of susceptibles and infecteds, originally from Raggett (1982). The final rows indicates, for each pair of consecutive observations, the size of the statespace for evaluating the transition probability and the ρ value for the associated rate matrix

Time	0	0.5	1.0	1.5	2.0	2.5	3.0	4.0
S	254	235	201	153	121	110	97	83
I	7	14	22	29	20	8	8	0
d_{HCS}	-	261	946	2059	1387	289	197	346
d	-	245	867	1868	1308	282	181	240
ρ	-	101.5	171.4	217.1	170.1	83.1	53.6	106.3

Table 2 Timings for estimating the full log-likelihood (1000 repeats) and the log-likelihood for the jump from the initial to the final observation (20 repeats) for the Eyam data set, number of sparse vector-matrix multiplications for one repeat, and the accuracies of the estimates. Results are given for the method of Ho et al. (2018) (HCS), the `expAtv` function in the `expm` package, which uses the Krylov subspace techniques of Sidje (1998), the method of Al-Mohy and Higham (2011) (AMH), the uniformisation algorithm (Unif) and the scaling and squaring algorithm (SS). [1] The timing for SS on the jump likelihood was estimated from a single repeat

Algorithm	Full likelihood			Jump likelihood		
	Time (secs)	Mult	Accuracy	Time (secs)	Mult	Accuracy
HCS	45.3	–	5.7×10^{-8}	9.7	–	4.3×10^{-9}
expAtv	558.5	–	1.6×10^{-10}	323.2	–	8.2×10^{-11}
AMH	–	3701	$< 1 \times 10^{-15}$	-	14300	$< 4 \times 10^{-14}$
Unif	18.72	1596	$< 1 \times 10^{-15}$	15.2	3921	$< 6 \times 10^{-14}$
SS	1678	–	1.1×10^{-13}	8940[1]	-	$< 6 \times 10^{-14}$

of the likelihood (20 estimates for the likelihood for the jump from $t = 0$ to $t = 4$). We also recorded the error in the evaluation of the log likelihood. Since for uniformisation, using renormalisation and two-tailed truncation together produced the fastest and most accurate evaluations, we only considered this combination. Given that the true likelihood is not known, the error using uniformisation, from scaling and squaring and from Al-Mohy and Higham (2011) were approximately bounded by examining their discrepancy from each other. The results are presented in Table 2.

Scaling and squaring is extremely slow in these high-dimensional scenarios; however, Sherlock and Golightly (2019) provides a bistable example, the Schlögel model, where $d \approx 100-200$ but $\rho > 10^5$, and the scaling and squaring algorithm outperforms uniformisation by orders of magnitude.

Since $m = \mathcal{O}(\rho)$ the choice of tolerance, ϵ, typically has only a small effect on the speed of the uniformisation algorithm. For the full likelihood evaluation, uniformisation is over twice as fast as the algorithm of Ho et al. (2018) and approximately thirty times as fast as `expAtv`, and is more accurate than either; it is also over twice as fast as the algorithm of Al-Mohy and Higham (2011), although both are very accurate.

For the single large jump between observations, we see the same pattern in terms of accuracy. There is a gain in efficiency by using two-tailed-truncation because ρ is

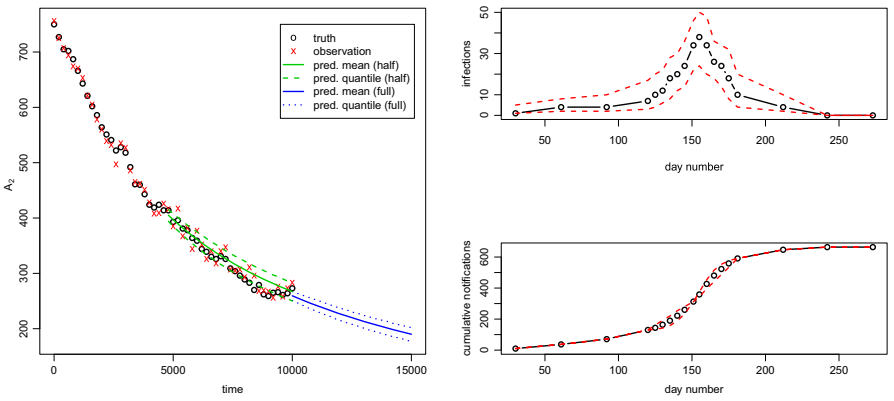

Fig. 1 Moran model (left): true values (o), observations (x), filtering/prediction mean (solid lines) and 95% quantiles (dashed and dotted lines) for a further time of 5000 from data up to $T = 5000$ and data up to $T = 10000$. Swansea measles SIR model (right): posterior median (solid line, with o to show the positions) and posterior 95% quantiles for the number of infected people at each real or latent observation time (top) and the cumulative number recovered by that time (bottom)

larger ($m_{lo} = 3081$ and $m = 3797$), but despite this, the method of Ho et al. (2018) is now more efficient than uniformisation, although considerably less accurate than it. Again, `expAtv` is over twenty times slower than uniformisation and less accurate, and AMH is over three times slower than uniformisation.

4.2 Maximum likelihood inference, filtering and prediction

We now consider the Moran model, which has four unknown parameters: (α, β, u, v) and $n_{pop} = 1000$. Setting $(\alpha, \beta, u, v) = (1, 0.3, 0.2, 0.1)$, we simulate a path of the process for $T = 10000$ time units. We then sample 51 observations at times $0, 200, 400, \ldots, 10000$, by taking the value of the process at each of these times and adding independent noise with a distribution of $\mathsf{Bin}(800, 0.5) - 400$.

We then perform inference on $\theta = (\log\alpha, \log\beta, \log[u/(1-u)], \log[v/(1-v)])$ by maximising the likelihood based on all the data and, separately, based on the data up to $T = 5000$. In each of these two data scenarios we find the filtering distribution, $\mathbb{P}\left(X_T | y_{0:T}, \widehat{\theta}\right)$, at time T via (2); finally we predict forward from T in steps of 200 for a further time of $T_{pred} = 5000$ by repeatedly multiplying the current distribution vector by $\exp(200\mathsf{Q}(\widehat{\theta}))$. The true values, observations and filtering and prediction distributions are shown in Fig. 1. The whole process of inference and prediction took less than two minutes on a single i7-3770 CPU running at 3.40GHz. Further, after defining Q, only 10 lines of `R` code are required to calculate the log-likelihood, and fewer than this to produce the filtering distribution (see Appendix E).

4.3 Bayesian inference for the Swansea measles epidemic of 2013

The largest measles outbreak in the United Kingdom between 2011 and 2019 centred around Swansea, Wales and occurred between November 2012 and July 2013. Of the

Table 3 Number of measles notifications in the Swansea Local Authority area by month (from http://www.wales.nhs.uk/sitesplus/888/page/66389, February 10th 2020)

Month	2012			2013						
	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
Day number	0	30	61	92	120	151	181	212	242	273
Notifications	0	10	27	34	59	183	278	56	17	0

1219 cases in mid- and west-Wales, 664 occurred in the Swansea Local Authority (LA) area, 243 in the nearby Neath and Port Talbot LA and fewer than 80 occurred in any of the other individual LA areas in West or South Wales (http://www.wales.nhs.uk/sitesplus/888/page/66389, accessed February 10th 2020). A reduction in uptake of the MMR (Measles, Mumps, Rubella) vaccine has been blamed e.g. Jakab and Salisbury (2013) for this, with particularly low rates reported in Swansea (https://en.wikipedia.org/wiki/2013_Swansea_measles_epidemic, accessed February 10th 2020).

The basic reproduction number, R_0, is the expected number of secondary infections in a susceptible population that arise directly from the primary infection of a single individual. For the SIR model described in Section 1.1, $R_0 = \beta/\gamma$. For measles, R_0 is often reported as between 14 and 18 (e.g. Anderson and May (1982)), which fits with the World Health Organisation (WHO) recommendation of vaccination level of at least $93 - 95\%$ WHO (2009).

We fit the SIR model to the notification data for the Swansea LA provided in Table 3 so as to estimate the overall R_0 for the partially vaccinated population in Swansea and to demonstrate inference on the unknown number of infectious individuals at each observation time. In fitting the model we are making several assumptions and simplifications, including the following. Firstly, we are ignoring infections from Swansea to other LAs and from these LAs to Swansea; since most of the infections occurred in Swansea the former will outnumber the latter and so we will underestimate the 'true' R_0, and provide a 'local' R_0 at the epicentre of the infection. Secondly it is known that the lowest level of vaccination, and the highest level of infection was amongst 10-18 year olds, see Wise (2013); a more accurate model would, therefore, partition the population into age groups. Age-stratified, continuous-time Markov chain SIR models are difficult to fit in general, however, and often a deterministic version of the model is used (e.g. Broadfoot and Keeling (2015)). Finally, we treat a notification as equivalent to a removal: this is not unreasonable as once an individual has been diagnosed by a GP with suspected measles they will be asked to isolate themselves.

As described in Section 2.2 we add as latent variables the number of infections at each of the reporting times, Days 30, 61, 92, ..., 212. The number of infections at times 242 and 273 must both be zero.

To understand the evolution near the peak of the epidemic and speed up inference still further, we add latent observation times during the peak of the infection, at Days 125, 130, 135, 140, 145, 155, 160, 165, 170 and 175. This leads to 10 further latent observations of the number of infected individuals and (because of constraints) 10

further latent observations of the number removed during each reduced time period, leading to a total of 27 integer

latent variables.

We use a $\mathsf{N}(\log 5, 2/3)$ prior for $\log R_0 = \log(\beta/\gamma)$, a $\mathsf{N}(\log(1/15), 1)$ prior for $\log \gamma$ and, because it is very poorly identified, we set the prior for the effective population size to $p(N_{pop} = n) \propto \exp(-n/500)1_{\{n \geq 1000\}}$.

We perform inference via a Metropolis-within-Gibbs algorithm: $\theta = (\log \beta, \log \gamma)$ is updated via a random walk proposal with a jump of $\mathsf{N}(0, \lambda^2 \mathsf{I}_2)$, n_{pop} via an integer-valued random walk proposal, and x_{latent}, the latent observations via integer random walks, with physical constraints (such as the sum of all the Rs not being able to exceed n_{pop}) checked for automatically; see Appendix D for more details.

The basic reproduction number, R_0, is estimated as 1.15, with a 95% credible interval of (1.01, 1.31). This fits with other information known: firstly,up until 2013, R_0 only changed gradually over time (due to year-on-year variations in infant vaccination rates) and it cannot have reached much higher than 1 in late 2012 as otherwise there would have been an outbreak in a previous year; secondly an R_0 of 1.15 if the true R_0 is 16, corresponds to a vaccination level of 93%, and $R_0 = 1.3$ corresponds to a 92% level, and as argued earlier, we expect to slightly underestimate R_0. As of December 2012, the estimated coverage of one dose of MMR vaccine among 16 year-olds in Wales was 91%; see Public Health Wales (2013).

The right-hand panels of Fig. 1 show the posterior median and 95% credible intervals for the number of infections at each of the monthly observation times and at the 10 additional latent times, and similar intervals for the cumulative number of infections. In any infectious disease, at any current time point, it is vital to understand the current, unknown, number of infections in order to be able to predict the future course of an epidemic.

5 Discussion

We have shown that inference, prediction and filtering for continuous-time Markov chains with a large but finite statespace, especially those arising from reaction networks is not just conceptually straightforward when the matrix exponential is used, but it is also often practical. We have provided and demonstrated the use of robust tools for this purpose in R, which opens up the direct use of and inference for reaction-network models to a wider audience. Straightforward inference for epidemic models, such as the SIR and SEIR models is particularly apposite at the time of submission, as it might have enabled an analysis of early COVID-19 infection data by people not expert in the more complex MCMC methodology typically used.

We emphasise that we are not suggesting that the tools we provide should replace the particle MCMC, ABC and SMC methods currently employed. In our experience, inference for epidemic models coded in a fast, compiled language is often more efficient in terms of effective samples per second, for example, than the approach using matrix exponentiation. However, the matrix exponential approach *is* much more straightforward, and the code that uses it can be written in the simpler, interpreted language R.

As the size of the statespace increases, the efficiency of the matrix exponentiation approach decreases; however, once the statespace becomes sufficiently large, the evolution of the process is often approximated by a stochastic differential equation (e.g. Golightly and Wilkinson 2005, Fearnhead et al. 2014) or, when the behaviour is effectively deterministic, by ordinary differential equations (e.g. Broadfoot and Keeling 2015).

For the scaling and squaring approach, in particular, the cost of the exponentiation of Q/K can be nearly halved by using a Padé approximant (e.g. Moler and Van Loan (2003)), but this then requires a matrix inversion, and so, for reasons of robustness, was not pursued here.

Acknowledgements I would like to thank Prof. Lam Ho for suggesting that the reformulation of the statespace in Ho et al. (2018) in terms of births might be applicable within the methodology presented herein. I am also grateful to Dr. Andrew Golightly for several useful discussions.

A Evaluating $m_\epsilon(\rho)$

Our fast, robust and accurate method for evaluating $m_\epsilon(\rho)$, as defined in Sect. 3.1 relies on the following new result.

Theorem 1 *If $\rho \leq \epsilon$, $m_\epsilon(\rho) = 0$, and if $\rho \leq \epsilon^{1/2}$, $0 \leq m_\epsilon(\rho) \leq 1$. More generally: $m_\epsilon(\rho) \leq \lceil m_+ \rceil$, where*

$$m_+ := \rho - \frac{1}{3}\log\epsilon \left\{1 + \left(1 - \frac{18\rho}{\log\epsilon}\right)^{1/2}\right\} - 1. \tag{8}$$

Furthermore,

$$\lfloor m_- \rfloor \leq m_\epsilon(\rho) \leq \lceil m_{++} \rceil,$$

where both inequalities require $\epsilon < 0.04$ and the latter also requires $\epsilon < 1 - e^{-\rho}$ and $B > \log\epsilon$, where

$$m_- := \rho + \{2\rho\}^{1/2}\left\{-\log(\epsilon\sqrt{2\pi}) - \frac{3}{2}\log A + \log(A-1)\right\}^{1/2}, \tag{9}$$

$$m_{++} := \rho + \frac{B - \log\epsilon}{3}\left\{1 + \left(1 + \frac{18\rho}{B - \log\epsilon}\right)^{1/2}\right\},$$

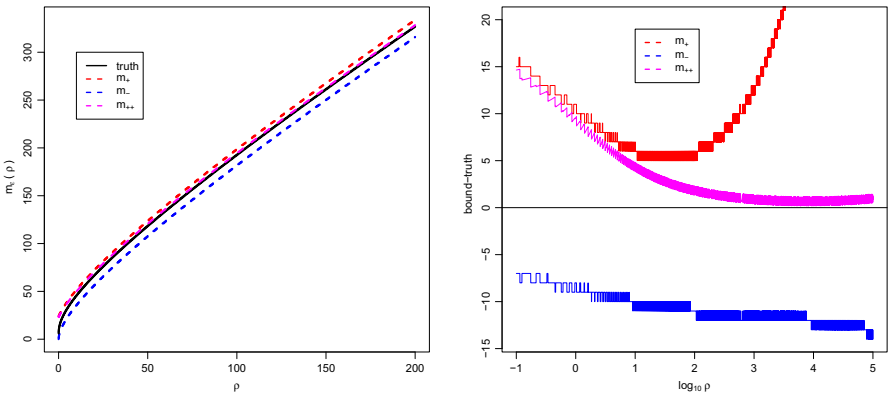

Fig. 2 Left panel: $m_\epsilon(\rho)$ together with its upper and lower bounds from Theorem 1, plotted against ρ for $\epsilon = 10^{-16}$. Right panel $\mathsf{bound}(\rho) - m_\epsilon(\rho)$ against $\log_{10}\rho$ for $\epsilon = 10^{-16}$

$$A := 2\rho h\left(\frac{m_+ + 1}{\rho}\right) \quad \textit{and} \quad B := -\frac{1}{2}\log 4\pi\rho h\left(\frac{m_-}{\rho}\right), \tag{10}$$

and $h(x) = x - 1 + x\log x$.

The bound (8) arises from a standard argument, whereas those in (9) and (10) are derived from extremely sharp but intractable bounds on $r_m(\rho) := \mathbb{P}(\mathsf{Poisson}(\rho) > m)$ in Short (2013); our bounds use only elementary functions and so are much quicker to compute than the quantile upper bound in Short (2013), yet from Fig. 2 they are still very sharp. The bounds in (9) and (10) together with the alternative form in (11)

$$r_m(\rho) = \frac{1}{\Gamma(m+1)}\int_0^\rho x^m e^{-x}\mathrm{d}x, \tag{11}$$

which follows from the equivalence between at least $m+1$ events of a Poisson process with a unit rate occuring by time ρ and the time until the $m+1$th event being at most ρ, permit a simple but fast binary search for $m_\epsilon(\rho)$.

A.1 Implementation details

Our binary search algorithm homes in on the required m using the upper and lower bounds of Theorem 1 together with the identity (11), the right hand side of which can be evaluated quickly and accurately using the standard `C++` toolbox, `boost`. This is quicker than the standard implementation of the Poisson quantile function (e.g. as implemented in `boost`), which uses the Cornish-Fisher expansion to approximate the quantile (hence needing an expensive evaluation of Φ^{-1}) and then conducts a local search.

A.2 Proof of Theorem 1

The simple bounds for small ρ arise because $e^{-\rho} > 1-\rho$. Hence $r_0(\rho) = 1-e^{-\rho} < \rho$ and if $\rho \leq \epsilon$, $m_\epsilon(\rho) = 0$. Furthermore, $r_1(\rho) = 1 - e^{-\rho}(1+\rho) < \rho^2$, so if $\rho \leq \sqrt{\epsilon}$ then $r_1(\rho) \leq \epsilon$, so $m_\epsilon(\rho) \leq 1$.

The other bounds all use aspects of the following result.

Lemma 1 *Let $h(x) := 1 - x + x\log x$, then for $x \geq 1$,*

$$\frac{3}{6+2(x-1)}(x-1)^2 \leq h(x) \leq \frac{1}{2}(x-1)^2.$$

Proof The left hand inequality holds for $x \geq 0$ and is from Boucheron et al. (2013) page 36. For the right hand inequality, set $g(x) = (x-1)^2/2$ and notice that $0 = \mathsf{h}(1) = g(1) = \mathsf{h}'(1) = g'(1)$, and $\mathsf{h}''(x) = 1/x \leq 1 = g''(x)$ for $x \geq 1$. □

The first upper bound on $m_\epsilon(\rho)$, (8), arises from a standard Chernoff argument (e.g. Boucheron et al. (2013)) to the right tail of a $\mathsf{Poisson}(\rho)$ random variable, X. The moment generating function is $M_X(t) = \mathbb{E}\left[e^{Xt}\right] = \exp[\rho(e^t - 1)]$, and by Markov's inequality:

$$\mathbb{P}(X \geq m) = \mathbb{P}\left(e^{Xt} \geq e^{mt}\right) \leq e^{-mt} M_X(t) = e^{-mt+\rho(e^t-1)}.$$

The inequality holds for all t and the right-hand side is minimised at $t = \log(m/\rho)$, giving

$$\mathbb{P}(X \geq m) \leq \exp[-\rho\mathsf{h}(m/\rho)] \leq \exp\left[-\rho\frac{3(m/\rho-1)^2}{6+2(m/\rho-1)}\right]$$

by Lemma 1. Setting $\epsilon = \mathbb{P}(X \geq m+1)$ and $y = (m+1)/\rho - 1$ gives $3\rho y^2(6+2y)\log\epsilon \geq 0$, from which $y \geq -\log\epsilon \times \sqrt{1 - 18\rho/\log\epsilon}/(3\rho)$, and (8) follows on substituting for y.

The much tighter bounds in (9) and (10) use Theorem 2 of Short (2013), which can be rewritten to state that

$$\Phi\left(-\sqrt{2\rho\mathsf{h}(m'/\rho)}\right) < \mathbb{P}(X > m) < \Phi\left(-\sqrt{2\rho\mathsf{h}(m/\rho)}\right), \tag{12}$$

where $m' := m+1$ and where the left hand side holds provided $m' > \rho$ and the right hand side holds provided $m > \rho$. We first show that these conditions are satisfied. Firstly, when $\rho < 1$, clearly $m' > \rho$, moreover $r_0(\rho) = 1 - e^{-\rho}$, so provided $1 - e^{-\rho} > \epsilon$, we require $m \geq 1 > \rho$. When $\rho \geq 1$, we use the easily verified facts that $r_m(m)$ is an increasing function of m and $r_m(\rho)$ is an increasing function of ρ; thus for $\rho \geq m \geq 1$, $r_m(\rho) \geq r_m(m) \geq r_1(1) = 1 - 2e^{-1} > 0.04$, and the tolerance condition is not satisfied. We, therefore need $m > \rho$ (which also gives $m' > \rho$).

Neither Φ^{-1} nor h^{-1} is tractable (functions that perform $\Phi^{-1}(p)$ solve $\Phi(x) = p$ iteratively), and even with the bounds on h from Lemma 1 and standard bounds on Φ

in terms of ϕ, tractable inversion is still not possible. We use the bound (8) to create (9), and then (9) to create (10).

To prove (9), since $\epsilon \leq 0.04$, from the left inequality in (12),

$$0.04 \geq \mathbb{P}(X \geq m) \Rightarrow \sqrt{2\rho \mathsf{h}(m'/\rho)} \geq -\Phi^{-1}(\epsilon) \approx 1.75 > \sqrt{3}.$$

Firstly, since $m_+ + 1 \geq m + 1$, this ensures $A > 1$, so $\log(A-1)$ is real. More importantly, it ensures that $[2\rho \mathsf{h}(m'/\rho)]^{-1/2} - [2\rho \mathsf{h}(m'/\rho)]^{-3/2}$ is a decreasing function of $[2\rho \mathsf{h}(m'/\rho)]^{1/2}$ and, since $\mathsf{h}'(x) > 0$ for $x > 1$, it is also a decreasing function of m'. The m' that we desire satisfies $m' \leq m_+ + 1 =: m'_+$, and hence

$$[2\rho h(m'/\rho)]^{-1/2} - [2\rho h(m'/\rho)]^{-3/2} \geq [2\rho h(m'_+/\rho)]^{-1/2} - [2\rho h(m'_+/\rho)]^{-3/2}.$$

Since, for $y > 0$, $\Phi(-y) > (1/y - 1/y^3)\phi(y)$,

$$\Phi\left(-\sqrt{2\rho h(m'/\rho)}\right) \geq \left\{[2\rho h(m'_+/\rho)]^{-1/2} - [2\rho h(m'_+/\rho)]^{-3/2}\right\} \phi\left(\sqrt{2\rho h(m'/\rho)}\right).$$

Combining the left inequality in (12) with the right-hand inequality in Lemma 1 gives

$$\epsilon \geq \frac{1}{\sqrt{2\pi}} \left\{[2\rho h(m'_+/\rho)]^{-1/2} - [2\rho h(m'_+/\rho)]^{-3/2}\right\} \exp\left[-\frac{(m'-\rho)^2}{2\rho}\right].$$

Equation (9) follows on rearrangement.

To show (10) we apply the right hand inequality in (12) and the bound $\Phi(-x) < \phi(x)/x$, then the fact that $m \geq m_-$, and finally Lemma 1 to find:

$$\begin{aligned}
\mathbb{P}(X > m) &< \frac{1}{\{4\pi\rho \mathsf{h}(m/\rho)\}^{1/2}} \exp\left[-\rho \mathsf{h}(m/\rho)\right] \\
&\leq \frac{1}{\{4\pi\rho \mathsf{h}(m_-/\rho)\}^{1/2}} \exp\left[-\rho \mathsf{h}(m/\rho)\right] \\
&\leq \frac{1}{\{4\pi\rho \mathsf{h}(m_-/\rho)\}^{1/2}} \exp\left[-3\rho \frac{(x-1)^2}{6+2(x-1)}\right],
\end{aligned}$$

where $x = m/\rho$. We must, therefore, ensure that the final bound is no more than ϵ. Rearranging this gives $3\rho(x-1)^2 - 2(B - \log\epsilon)(x-1) - 6(B - \log\epsilon) \leq 0$, so that when $B - \log\epsilon > 0$, $x - 1 \leq (B - \log\epsilon)(1 + \sqrt{1 + 18\rho/(B - \epsilon)})/(3\rho)$.

B Scaling and squaring: choosing *s*

Recall that the scaling and squaring algorithm evaluates $\exp(\mathsf{M})$ via the equality $\exp(\mathsf{M}) = \left[\exp(\mathsf{M}/2^s)\right]^{2^s}$. Calculation of $\exp(\mathsf{M}/2^s)$ takes $m_\epsilon(\rho/2^s)$ matrix-matrix

multiplications; repeatedly squaring this quantity s times takes s matrix-matrix multiplications, leading to a total cost of

$$c(s) = m_\epsilon(\rho/2^s) + s.$$

Asymptotically in large ρ, $\mathsf{Pois}(\rho)/\rho \sim 1$, so $m_\epsilon(\rho) \approx \rho$. With $m_\epsilon(\rho/2^s) = \rho/2^s$, $c(s)$ is minimised at

$$\widehat{s_1} = \frac{\log\rho + \log\log 2}{\log 2}.$$

Since $m_\epsilon(\rho)$ is an upper quantile, we also have that $m_\epsilon(\rho) > \rho$, and, for low ρ, we find $m_\epsilon(\rho) \gg \rho$. Indeed, $m_\epsilon(\rho/2^s)$ does not, in fact, decrease as quickly as $\rho/2^s$, so

$$c'(s) = \frac{\mathrm{d}}{\mathrm{d}s} m_\epsilon(\rho/2^s) + 1 \geq \frac{\mathrm{d}}{\mathrm{d}s}\frac{\rho}{2^s} + 1.$$

Since the gradients both become less negative as s increases (indeed, they asymptote to 0), $\widehat{s_1}$ is a strict lower bound on the true minimum $\widehat{s}$. For a range of ϵ values from 0.1 to 10^{-16}, and a range of ρ values from 0.1 to 10^6, we have found that the true optimum s always lies in the range between $\widehat{s_1}$ and $\widehat{s_1} + 6$. When the scaling and squaring algorithm is required, $\widehat{s}$ chosen as the integer argument within this range that minimises $c(s)$.

When M is dense, the above algorithm finds the optimal choice of s; however, when M is sparse, the matrix multiplications required to evaluate $\exp(\mathsf{M}/2^s)$ are cheaper than those involved in the subsequent squaring. Hence, the cost is minimised at a slightly lower s, which depends on the sparsity of M (as well as ρ and ϵ). For simplicity, we set $\widehat{s} \leftarrow \widehat{s} - \min(2, \widehat{s})$.

C Functions in the `expQ` package

The functions in the `expQ` package are provided below. Each function requires a rate matrix, Q, which can be sparse or dense, and a precision, ϵ. `Unif_v_exp_Q` takes a horizontal vector, v, and calculates $v \exp \mathsf{Q}$ via uniformisation. `SS_v_exp_Q` takes a horizontal vector, v, and calculates $v \exp \mathsf{Q}$ via scaling and squaring. `v_exp_Q` takes a horizontal vector, v, and calculates $v \exp \mathsf{Q}$ via whichever is likely (based on empirical results on an i7-3770 CPU) to be the more efficient of uniformisation or scaling and squaring. `vT_exp_Q` takes a vertical vector, v, and calculates $\left(v^\top \exp \mathsf{Q}\right)^\top$ via whichever is likely (based on empirical results on an i7-3770 CPU) to be the more efficient of uniformisation or scaling and squaring. `SS_exp_Q` calculates $\exp \mathsf{Q}$ using scaling and squaring.

D Latent-variable updates for the SIR model

Our particular reduced-statespace implementation of the SIR model fit for the Swansea Measles epidemic uses 10 additional latent observation times, 5 between days 120 and 151 (at days 125, 130, 135, 140 and 145) and five between days 151 and 212 (at days 155, 160, 165, 170 and 175). This leads to 27 latent variables: 17 unknown number of infecteds at the (true and latent) observation times and 10 (not 12 because two sums are known) unknown numbers of recovered for the time period since the previous (true or latent) observation time. We emphasise that the R latent variables are *not* the cumulative number of recovered individuals since the epidemic began.

When a new latent vector is proposed, we first check whether it can possibly fit with the current n_{pop} and the known data. If it does not fit, then the proposal may be rejected without any matrix exponentiation. At the jth (true or latent) time point, denote the current number of infecteds by I_j and the number removed since the previous time point by R_j. Let J be the total number of (true and latent) time points. Note that $S_j = n_{pop} - I_0 - I_j - \sum_{i=1}^{j} R_i$. The following checks are performed:

1. For each $j = 1, \ldots, J$: $I_j \geq 0$, $R_j \geq 0$ and $S_j \geq 0$.
2. For each $j = 1, \ldots, J$: $S_j \leq S_{j-1}$.
3. For each $j = 1, \ldots, J-1$: $I_j \leq \sum_{i=j+1}^{J} R_i$.

The third constraint arises because there are no active infections at the end of the epidemic. These constraints can hinder the mixing of the integer-valued random walk algorithm on the latent variables, so we split the latent variables into four groups, grouped by observation time. This grouping has the additional advantage that only a subset of matrix exponentiation calculations need be performed for each of the four individual proposals.

E Log-likelihood R code for the Moran model

To demonstrate the simplicity of inference via the matrix exponential, we provide code to evaluate the log-likelihood for the Moran model. Code for the filtering distribution is very similar but there is no need to track the re-normalisation constant (in `ll`).

```
## Log likelihood for Moran model
## thetaunk=(log alpha, log beta, logit u, logit v)
## npop=known population size
## obstim=vector of observation times
## yobs=vector of observations
## errn=parameter for Binom(2*errn,0.5)-errn error distribution
## nu=t(rep(1/d,d)); ## uniform prior over statespace
getll<-function(thetaunk,npop,obstim,yobs,errn,nu) {
    thetas=c(exp(thetaunk[1:2]),exp(thetaunk[3:4])/(1+exp(thetaunk[3:4])),npop)
    nobs=length(obstim)
    d=npop+1 ## size of statespace; states are 0, ..., npop

    Q=MoranGetQ(thetas) ## same Q every time as whole statespace
    ll=0
    nu=nu*dbinom(yobs[1]+errn-(0:npop),2*errn,0.5)
    for (i in 2:(nobs-1)) {
        currtot=sum(nu)
```

```
        if ((currtot<1e-6)||(currtot>1e6)) { ## avert possible over/underflow
            nu=nu/currtot
            ll=ll+log(currtot)
        }
        nu=Unif_v_exp_Q(nu,Q*(obstim[i+1]-obstim[i]),1e-15)
        nu=t(as.vector(nu)*dbinom(yobs[i+1]+errn-(0:npop),2*errn,0.5))
    }
    ll=ll+log(sum(nu))

    return(ll)
}
```

References

Al-Mohy AH, Higham NJ (2011) Computing the action of a matrix exponential with an application to exponential integrators. SIAM J. Sci. Comput. 33(2):488–511

Amoros R, King R, Toyoda H, Kumada T, Johnson PJ, Bird TG (2019) A continuous-time hidden Markov model for cancer surveillance using serum biomarkers with application to hepatocellular carcinoma. METRON 77:67–86

Anderson RM, May RM (1982) Directly transmitted infectious diseases: control by vaccination. Science 215:1053–1060

Andersson H, Britton T (2000) Stochastic epidemic models and their statistical analysis. Springer, New York

Andrieu C, Doucet A, Holenstein R (2009) Particle Markov chain Monte Carlo for efficient numerical simulation. In: L'Ecuyer P, Owen AB (eds) Monte Carlo and Quasi-Monte Carlo Methods 2008. Spinger, Berlin, pp 45–60

Andrieu C, Doucet A, Holenstein R (2010) Particle Markov chain Monte Carlo methods. J. R. Statist. Soc. B 72(3):269–342

Black AJ (2019) Importance sampling for partially observed temporal epidemic models. Stat Comput 29(4):617–630

Boucheron S, Lugosi G, Massart P (2013) Concentration inequalities?: a nonasymptotic theory of independence. Oxford University Press, Oxford

Broadfoot K, Keeling M (2015) Measles epidemics in vaccinated populations. Accessed on 1 Feb 2020

Doucet A, Pitt MK, Kohn R (2015) Efficient implementation of Markov chain Monte Carlo when using an unbiased likelihood estimator. Biometrika 102:295–313

Drovandi CC, McCutchan R (2016) Alive SMC^2: Bayesian model selction for low-count time series models with intractable likelihoods. Biometrics 72:344–353

Eddelbuettel D, Sanderson C (2014) Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. Comput. Stat. Data Anal. 71:1054–1063

Fearnhead P, Giagos V, Sherlock C (2014) Inference for reaction networks using the Linear Noise Approximation. Biometrics 70:457–466

Gallopoulos E, Saad Y (1992) Efficient solution of parabolic equations by Krylov approximation methods. J. Appl. Stat. 13(5):1236–1264

Georgoulas A, Hillston J, Sanguinetti G (2017) Unbiased bayesian inference for population markov jump processes via random truncations. Stat. Comput. 27(4):991–1002

Golightly A, Sherlock C (2019) Efficient sampling of conditioned Markov jump processes. Stat. Comput. 29(5):1149–1163

Golightly A, Wilkinson DJ (2005) Bayesian inference for stochastic kinetic models using a diffusion approximation. Biometrics 61(3):781–788

Golightly A, Wilkinson DJ (2011) Bayesian parameter inference for stochastic biochemical network models using particle Markov chain Monte Carlo. Interface Focus 1(6):807–820

Golightly A, Wilkinson DJ (2015) Bayesian inference for Markov jump processes with informative observations. Stat. Appl. Genet. Mol. Biol. 14(2):169–188

Ho LST, Crawford FW, Suchard MA (2018) Direct likelihood-based inference for discretely observed stochastic compartmental models of infectious disease. Ann. Appl. Stat. 12(3):1993–2021

Jakab Z, Salisbury DM (2013) Back to basics: the miracle and tragedy of measles vaccine. The Lancet 381:1433–1434
Jenkinson G, Goutsias J (2012) Numerical integration of the master equation in some models of stochastic epidemiology. PLOS ONE 7(5):1–9
Kinyanjui T, Middleton J, Güttel S, Cassell J, Ross J, House T (2018) Scabies in residential care homes: Modelling, inference and interventions for well-connected population sub-units. PLOS Comput. Biol. 14(3):1–24
Koblents E, Miguez J (2015) A population Monte Carlo scheme with transformed weights and its application to stochastic kinetic models. Stat. Comput. 25(2):407–425
Kypraios T, Neal P, Prangle D (2017) A tutorial introduction to bayesian inference for stochastic epidemic models using approximate bayesian computation. Math. Biosci. 287:42–53
McKinley TJ, Ross JV, Deardon R, Cook AR (2014) Simulation-based Bayesian inference for epidemic models. Comput. Stat. Data Anal. 71:434–447
Moler C, Van Loan C (2003) Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Rev. 45(1):3–49
Moran P (1958) Random processes in genetics. Math. Proc. Camb. Philos. Soc. 54(1):60–71
Norris JR (1997) Markov chains. Cambridge University Press, Cambridge
Owen J, Wilkinson DJ, Gillespie CS (2015) Likelihood free inference for Markov processes: a comparison. Stat. Appl. Gen. Mol. Biol. 14(2):189–209
Public Health Wales (2013) Vaccine uptake in children in wales: October to december 2012
R Core Team (2018) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna
Raggett G (1982) A stochastic model of the Eyam plague. J. Appl. Stat. 9(2):212–225
Rao V, Teh YW (2013) Fast mcmc sampling for markov jump processes and extensions. J. Mach. Learn. Res. 14:3295–3320
Reibman A, Trivedi K (1988) Numerical transient analysis of markov models. Comput. Oper. Res. 15(1):19–36
Ross SM (1996) Stochastic processes. Wiley, New York
Saad Y (1992) Analysis of some Krylov subspace approximations to the matrix exponential operator. J. Appl. Stat. 29(1):209–228
Sanderson C, Curtin R (2016) Armadillo: a template-based C++ library for linear algebra. J. Open Source Softw. 1:26
Sanderson C, Curtin R (2018) A user-friendly hybrid sparse matrix class in C++. LNCS 10931:422–430
Sherlock C,Golightly A (2019). Exact bayesian inference for discretely observed markov jump processes using finite rate matrices
Sherlock C, Thiery A, Roberts GO, Rosenthal JS (2015) On the effciency of pseudo-marginal random walk Metropolis algorithms. Ann. Stat. 43(1):238–275
Short M (2013) Improved inequalities for the Poisson and binomial distribution and upper tail quantile functions. ISRN Probabil. Stat. 2013(3):1–6
Sidje RB (1998) EXPOKIT: a software package for computing matrix exponentials. ACM Trans. Math. Soft. 24(1):130–156
Sidje RB, Stewart WJ (1999) A numerical study of large sparse matrix exponentials arising in Markov chains. Comput. Stat. Data Anal. 29(3):345–368
Vellela M, Qian H (2009) Stochastic dynamics and non-equilibrium thermodynamics of a bistable chemical system: the Schlögel model revisited. J. Royal Soc. Interface 6(39):925–940
WHO (2009) Who position on measles vaccines. Vaccine 27(52):7219–7221
Wilkinson DJ (2012) Stochastic modelling for systems biology. CRC Press, Boca Raton
Wise J (2013). Largest group of children affected by measles outbreak in wales is 10–18 year olds. *BMJ*, 346