Simulation-based Inference for Partially Observed Markov Process Models with Spatial Coupling

by

Kidus Asfaw

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Statistics)
in the University of Michigan
2021

Kidus Asfaw

kasfaw@umich.edu

ORCID iD: 0000-0001-8625-3743

Acknowledgements

There are so many people who have helped me reach this stage of my life's journey and I would like to take a moment to thank them.

I would first like to thank Dr. Edward Ionides whose advising has not only propelled me forwards throughout this program but has also been given in the most helpful and non-judgmental way possible. I appreciate the innumerable hours he has invested in my growth and our work together - it has been rewarding to work with someone who simply wants me to be great at what I do. He has been a brilliant adviser and also leads by example with his style of work and perspective on life. Thank you, Ed, for your support.

I would also like to thank Dr. Aaron King for spending so much time and effort helping me get through conceptual and technical problems. Aaron has made me feel welcome both in his lab group and in his family and I am so thankful for these soft touches. I would have been at the risk of never witnessing a UM basketball game were it not for his efforts to foster community.

My committee has been generous with their time and I would also like to thank Dr. Stilian Stoev for giving me advice in the very first days of course selection, teaching me STATS 620 and making himself available for any advising that I needed on this dissertation. I extend my gratitude to Dr. Jon Zelner for his guidance throughout the final stages of this program. I am also grateful for my graduate student colleagues, department staff and faculty who have helped me round out my training at the university.

I owe so very much to my partner Avery Williamson without whom this journey would not have been possible. I thank her for pushing me to be the best version of myself in work and in life. Her ambition and work habits have motivated me greatly; her jokes have made me laugh heartily; and her commitment to family and friends has moved me. I feel so lucky and grateful to have her in my life.

Another big thank you goes to my family - blood and chosen. Though I will never know for sure, my mother has probably spent as many late nights worrying about me in the last five years as I have spent awake with one concern or another. I thank her for her (suprisingly effective) prayers and for believing in me without any reservations. My brother's graduate school journey influenced so many of my decisions in this program and I feel so lucky to have someone who is blazing the trail that I can follow and I am grateful for his omnipresent helping hand. I am also glad he deals with anxiety in a more easy-going manner because my mother and I carry enough for a whole family. Thank you, Abe, and I love you. Indale, Mesfin, Meski and Teshe have been a constant sounding board when I needed it and I can only say thank you to this team of family members whose support has been constant throughout my life even when we were continents apart. I wouldn't be where I am today without them. Damaris and Michael have welcomed me into their homes and families with open arms. I thank them and Joe for providing me a home away from home during this program and I am eager to a life of continued communing and laughing with them. Iris, Marisa and Tom have made me feel like a fellow rugrat adventuring through Philly and beyond. How lucky I am to have them.

A deeply sincere tip of the hat goes to my neighbors and friends, Tahir and Su'ad. At times when self-doubt was holding me back, it has been reassuring to get comfort from neighbors that I respect and admire deeply. It has been an honor communing with you and I can only say thank you.

A final thanks goes to Mylah, who taught me to stay ready so that I do not have to get ready.

I have so many others to thank but I must leave it at this for now. Peace and love to everybody.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Statistical inference for nonlinear and non-Gaussian dynamic models of moderate and high dimensions is an open research area. Such models may require simulation-based methodology when linearization and Gaussian approximations are not appropriate. The *particle filter* has allowed likelihood-based inference for such problems when the dimension of the problem is small, but degrades in performance as the dimension of the model increases. This is due to the exponential growth in the volumes to be represented by Monte Carlo simulations as dimension grows. In epidemiology, this *curse of dimensionality* problem occurs when we jointly model the epidemiological dynamics in a group of neighboring towns that are coupled via immigration or travel. In this dissertation, I present two innovations that make methodological and practical progress in data analysis for nonlinear and non-Gaussian dynamic models with coupled disease models as the primary problem of interest. All work was done jointly with my co-advisers Dr. Ionides and Dr. King as well as Dr. Joonha Park and Allister Ho.

The first innovation is a group of simulation-based methods that take advantage of *localization* — the idea that dependence between far enough spatial units in a spatially coupled model is negligible — to make approximations enabling scalable likelihood estimation. I show theoretical results for the methods and examples of its use on three different models, including a coupled measles model in England.

The second innovation is the open-source R package **spatPomp**. This package builds on the strengths of the **pomp** package for model development and testing while adding new components that allow the implementation of new methods that are tailored for moderate- and high-dimensional problems. Various algorithms and utility functions are implemented and the package is available on the Comprehensive R Archive Network (CRAN) repository of packages.

Chapter 1

Introduction

1.1 Overview

In epidemiology and many other areas, we obtain data over time that give us partial information about a dynamic process like the spread of a disease in a population. The data can be used to support or discount candidate dynamic models for the process. Many epidemiological models are formulated as deterministic ordinary differential equations (ODEs). ODEs are natural starting points for modeling dynamic processes because they express the rates at which the states of the model system change over time and how they depend on other states. Solving ODEs gives us trajectories of the model system that can be compared to the data obtained from the dynamic process being modeled. Nevertheless, it can be desirable to fit stochastic models instead of ODEs. For instance, modeling transmission probabilistically allows us to explain why some viral introductions into a fully susceptible population do not lead to an outbreak (Keeling and Rohani, 2009). It can also allow us to achieve viral extinction in cases where the susceptible population is depleted whereas solutions to ODEs often approach zero asymptotically. Stochastic models can hence fit our data better. Further, a problem with deterministic models is that they imply a level of predictability which may not be justified. Unlike deterministic models, stochastic models allow us to express uncertainty in estimated model quantities.

One class of stochastic models that is used for time series data analysis is the partially observed Markov process (POMP) model class. Models in this class suppose the existence of a latent dynamic model which is Markovian, i.e., all the information about the latent state is independent of all historical latent state information conditional on the most recent latent state. The measurement model that outputs the observed data conditional on the latent states shadows the latent process and gives us information about the latent process. POMP models are convenient for modeling time series data because the separation between the latent process model and the measurement model allows the modeler to test model variations in either or both model components.

Given a POMP model, the information included in our data about the model is captured by the evaluation of a statistical function called the likelihood function. Models that are more consistent with the data have higher values of the likelihood function. Another quantity of interest for a POMP model is the distribution of the latent state at a given time conditional on data observed until that time. This distribution is called the *filter distribution.* Only rarely can we analytically compute an evaluation of the likelihood function or the filter distribution at a given time for a POMP model because they both involve integration over all possible latent states. Unless the POMP model involves latent process and measurement densities that are Gaussian or the state-space is finite and small, alternative methods must be found for estimating likelihoods and filter distributions.

In this dissertation we focus on a class of such methods that allows the user to provide an implicit model for the latent process by specifying a simulator for the latent process which can be used to perform numerical integration over simulated latent states. This is related to the concept of *mechanistic modeling*, which involves specifying a model for a dynamic system that describes the mechanism that drives the dynamics. Compartment models in epidemiology (e.g. Susceptible-Infected-Recovered, or SIR, models), which model transitions of people through different phases of disease status are an example of mechanistic models. Stochastic Euler solutions of mechanistic models can be used to simulate from the latent process. By providing Euler simulators for small time-increments, the user can then allow for statistical inference of the mechanistic model by using the approaches studied in this dissertation.

Methods for POMP models that only require a latent state transition simulator and a probability model for the conditional distribution for measurements at observation times given the latent states at those times are called *plug-and-play* (Bretó et al., 2009; He et al., 2010) methods. One such method that sequentially estimates the filter distribution at the observation times and the likelihood of the data is called the *particle filter* or *sequential Monte Carlo* (SMC) (Gordon et al., 1993). The particle filter has the property that it provides unbiased estimates of the likelihood of the data (Del Moral and Guionnet, 2001) and forms the basis for parameter estimation via iterated filtering with parameter perturbations (Ionides et al., 2006, 2015).

The particle filter involves resampling Monte Carlo simulations (also called *particles*) according to weights that signify the consistency of each particle to data. When the latent state space grows in dimension (which is usually accompanied by growing measurement dimension), the volume of the space to be represented by Monte Carlo simulations for numerical integration grows exponentially, which leads to massive imbalance in the particle weights. The resampling step of the particle filter then results in the representation of a high-dimensional space by a handful of particles (Bengtsson et al., 2008). The consequences of this *curse of dimensionality* (COD) phenomenon are that our high-dimensional filter distribution is approximated by very few particles and our likelihood estimates have high variance.

In epidemiology, the COD problem occurs when we consider a joint model of disease dynamics among, say, neighboring towns that have dependence via travel or immigration. Disease models for a small town may, for instance, predict that there will be viral extinction but a sustained epidemic could be observed due to continuous case importation from neighboring towns. In such cases, policy recommendations could be improved by fitting a joint model.

1.2 Previous approaches

In the previous example, one way to move forward when inference on a joint model is difficult would be to posit a model that treats the latent dynamics in the towns as independent with some shared model parameters (e.g. infectious period) and some spatial unit-specific parameters (e.g. town-specific reporting rate of cases) (Bretó et al., 2009). This allows us to use existing particle filter-based methods to analyze our data and retain the mechanistic nature of our model. However, approaches like this confine us to models without dynamic coupling that may compensate for model misspecification by outputting parameter estimates that do not correspond to the physical quantities they represent. For instance, suppose we have city-level time series data that comes from a coupled disease model in which the latent dynamics of different cities are dependent (perhaps via travel or immigration). If we fit a model that supposes independence between the city dynamics, the model may prefer to explain a sustained outbreak in a city by assigning likelihoods to transmission parameters that are higher than would be expected from understanding the science of the disease. The model that generated the data, however, may have used importation of infectives from other cities to sustain the outbreak.

The ensemble Kalman filter (EnKF) (Evensen and van Leeuwen, 1996) allows more flexibility by allowing us to specify latent state transition simulations that include coupling parameters that induce dependence between spatial units. However, EnKF is a sequential method which relies on an update step that assumes a Gaussian distribution for the latent state conditional on the data at each observation time. While this approximation makes EnKF scale well with growing dimension size, it can lead to bias (as shown in Chapter 4).

Outside of the plug-and-play class of methods, alternatives exist that involve making distributional assumptions about the latent process which make likelihood evaluation analytically tractable. The Kalman filter, for example, assumes a Gaussian transition density to provide the likelihood under a linear-Gaussian model. The likelihood of data from a Gaussian auto-regressive moving average (ARMA) process, for example, can be evaluated using the Kalman filter. EnKF can serve as a plug-and-play alternative in this case, up to Monte Carlo error, since we can supply a transition simulator as well as a transition density. The standard Expectation-Maximization (EM) algorithm is another method for evaluating likelihoods that requires an explicit transition density for the latent dynamics and is, hence, non-plug-and-play. Ultimately non-plug-and-play approaches make assumptions about the latent process that lead to analytical tractability of the likelihood evaluation problem at the risk of biased estimates. Simulation-based techniques allow the user to focus more on specifying the science of a disease and have an advantage when the disease dynamics are nonlinear and non-Gaussian.

1.3 Software as a home for methodology

The **pomp** package (King et al., 2016) was one of the outputs of a working group that focused on building inference methodology and software for, among other things, low-dimensional POMP models. Since its publication, the package has over 210 citations and is considered to be a major software package in R for statistical inference for low-dimensional nonlinear and non-Gaussian epidemiological models. As more attention is given to spatially coupled models, there has been a growing need for methods and tools that can help us mitigate the COD problem discussed above. Just as new methods that push existing boundaries are indispensable, so too are software tools that allow easy model implementation and criticism using new methods. What has allowed **pomp** to be successful in this respect?

First, the general abstraction of POMP models allows the user to implement various kinds of POMP models (e.g. discrete- or continuous-time latent models). Second, the user can easily provide components of the model only necessary for the methods to be used (e.g. specifying a transition density is not required to use plug-and-play methods like the particle filter). Third, a wide range of methods are implemented (Bayesian, frequentist, plug-and-play, non-plug-and-play, full-information, feature-based) which have computationally expensive pieces of code written in C. Fourth, extensive package documentation and case studies available from courses and modules make it easy to pick up as a new user. Finally, it implements example POMP models that allow the user to quickly test existing methods or even develop new ones.

The **spatPomp** package aims to emulate these advantages and be a home for statistical inference for POMP models with spatial coupling. There have been various methodological contributions from researchers that can enable complex model development and inference. Each method has its advantages and a class of problems that play to its advantages. Housing these approaches under one software roof and modularizing their component pieces allows us to quickly develop and test methods while giving the applied scientist a home for trying alternative methods.

1.4 Overview and contributions of this dissertation

1.4.1 Chapter 2

Estimation of the likelihood and the filter distribution can be accomplished by sampling from the latent process model and using the conditional density of the data given the sampled latent states. This is an example of a more general method called *importance sampling*, which samples from a target distribution by sampling from a different distribution called the *importance distribution* and attaching a *proper importance weight* to each sample from our importance distribution. In Chapter 2, I first introduce a method that uses the latent process model as the importance distribution but uses an approximation of the proper importance weight in order to approximately sample from the filter distribution and estimate the likelihood. I show that this method averts the COD. I then introduce a derivative method that can be more stable. Instead of using the latent process model, the so-called *adapted distribution* is used as the importance distribution. Since samples from this distribution are adapted to our data, they allow us to focus our computational effort in regions of the state space that are consistent with the data. However, sampling from this distribution using importance sampling poses a COD problem as does estimating the proper importance weight corresponding to it. The first problem is overcome by having independent bootstrap replicates describing their own regions of the state space for the importance sampling. The second is mitigated by using an approximation of the proper importance weight that uses a notion of localization in the time series data. I present a theoretical result for this method and illustrate all methods on a linear-Gaussian system and a nonlinear chaotic model.

1.4.2 Chapter 3

Spatiotemporal inference for nonlinear and non-Gaussian dynamics is an active research area that warrants a home for implementing new methodology and testing existing ones. Chapter 3 introduces the open-source **spatPomp** package, which retains the flexibility of the **pomp** package for model development while also extending its core functionalities to allow easy implementation of new methodology. New methods may, for instance, require the user to provide components of the measurement model other than the conditional density of the data given a latent state (like the conditional expectation or variance). The need for such model components spurred development of core **pomp** and **spatPomp** infrastructure that has allowed the implementation of several published methods that did not have available open-source code. The chapter illustrates how to use the core methods of the package on a linear-Gaussian system and also gives an example of how to develop a nonlinear disease transmission model.

1.4.3 Chapter 4

The methods introduced in Chapter 2 are intended for likelihood estimation. However, a rigorous examination of the following is required to better understand the methods:

- How do the methods scale with increasing dimension for a typical nonlinear and non-Gaussian disease transmission model?
- How do the tuning parameters affect the performance of the methods?

- Can these likelihood estimation algorithms be used for parameter estimation?
- How does the bias due to the approximations in the methods affect our inference task?

Chapter 4 provides the insights gained from extensive investigation of our methods on simulated measles data.

It is my hope that the work that led to this dissertation sustains its usefulness through the new methods and software that it produced. The immediate impact of the work is likely that there are now new methods that are able to push the limits of statistical inference for moderate-dimensional nonlinear and non-Gaussian problems. Given that this research area is still in its early stages, the new methods can be useful on some subset of problems, which is gratifying in itself. Perhaps a longer-term impact of the work is that **spatPomp** has eased the implementation and testing of new methods. I have learned a great deal about new and existing methods by implementing them in the package and conducting simulation studies. I describe two potential follow-up projects in my concluding chapter which arose from extensive simulation studies that were not previously available. I also show that a method that was not extensively used in applications, the block particle filter (Rebeschini and van Handel, 2015), can be quite useful and will likely form the basis of some future work among my collaborators. The signs are, therefore, that there is more to be done yet.

Chapter 2

Bagged Filters for Partially Observed Interacting Systems

2.1 Introduction

Bagging is a technique to improve numerically unstable estimators by combining an ensemble of replicated bootstrap calculations (Breiman, 1996). In the context of nonlinear partially observed dynamic systems, the bootstrap filter of Gordon et al. (1993) has led to a variety of particle filter (PF) methodologies (Doucet et al., 2001; Doucet and Johansen, 2011). We consider algorithms combining an ensemble of replicated particle filters, and we use the name *bagged filter*. Standard PF methods suffer from a curse of dimensionality (COD), defined as an exponential increase in computational requirement as the problem size grows, limiting its applicability to large systems (Bengtsson et al., 2008; Snyder et al., 2015; Rebeschini and van Handel, 2015). The COD presents empirically as numerical instability of the Monte Carlo algorithm for attainable numbers of particles. Much previous research has investigated scalable approaches to filtering and inference with applications to spatiotemporal systems. Our bagged filters are in the class of *plug-and-play* algorithms, meaning that they require as input a simulator for the latent dynamic process but not an evaluator of transition probabilities (Bretó et al., 2009; He et al., 2010). This simulation-based approach, also known as *likelihood-free* (Brehmer et al., 2020) or *equation-free* (Kevrekidis and Samaey, 2009), facilitates application to a wide class of models. The ensemble Kalman filter (Evensen and van Leeuwen, 1996; Lei et al., 2010; Katzfuss et al., 2020) is a widely used plug-and-play method which uses simulations to parameterize a Gaussian-inspired filtering rule. Another plug-and-play approach to combat the COD is the block particle filter (Rebeschini and van Handel, 2015; Ng et al., 2002). Both ensemble Kalman filter and block particle filter methods construct trajectories that can violate smoothness and conservation properties of the dynamic model. By contrast, our bagged filters are built using valid trajectories of the dynamic model, making localization approximations only when comparing these trajectories to data.

The replicated stochastic trajectories in a bagged filter form an ensemble of representations of the dynamic system. Unlike the particles in a particle filter or ensemble Kalman filter, the bagged replicates are independent in a Monte Carlo sense. Bagged filters therefore bear some resemblance to *poor man's ensemble* forecasting methodology in which a collection of independently constructed forecasts is generated using different models and methods (Ebert, 2001). Poor man's ensembles have sometimes been found to have greater forecasting skill than any one forecast (Leutbecher and Palmer, 2008; Palmer, 2002; Chandler, 2013). One explanation for this phenomenon is that even a hypothetically perfect model cannot provide effective filtering using methodology afflicted by the COD. We show that bagged filter methodology can relieve this limitation. From this perspective, the independence of the forecasts in the poor man's ensemble, rather than the diversity of model structures, may be the key to its success.

In the simplest bagged filter, each replicate simulates a realization of the latent process model. We call this the unadapted bagged filter (UBF) since the replicates in the ensemble depend on the model but not on the data. UBF is described in Sec. 2.2, with a theoretical analysis presented in Sec. 2.2.2. Each UBF replicate corresponds to a basic PF algorithm with a single particle. We show that UBF formally beats the COD under a weak mixing assumption, though UBF can have poor numerical behavior if a very large number of replicates are needed to reach this happy asymptotic limit. Subsequent empirical results show that UBF may nevertheless be a useful algorithm in some situations. In Sec. 2.3, we generalize UBF to construct an adapted bagged filter (ABF) where each replicate tracks the data. The price of adaptation is that ABF no longer fully avoids the COD, a limitation that can be controlled in certain situations by supplementing ABF with a technique called intermediate resampling, to obtain the ABF-IR algorithm. Theoretical results for ABF and ABF-IR algorithms are developed in Sec. 2.3.2. The algorithms are demonstrated in action and compared with alternative approaches in Sec. 2.4.

2.2 The unadapted bagged filter (UBF)

2.2.1 Notation

Suppose the collection of units is indexed by the set $\{1, 2, \ldots, U\}$, which is written as $1\!:\!U$. The latent Markov process is denoted by $\{\boldsymbol{X}_n, n \in 0\!:\!N\}$, with $\boldsymbol{X}_n = X_{1:U,n}$ taking values in a product space $\mathbb{X}^U$. This discrete time process may arise from a continuous time Markov process $\{\boldsymbol{X}(t), t_0 \leq t \leq t_N\}$ observed at times $t_{1:N}$, and in this case we set $\boldsymbol{X}_n = \boldsymbol{X}(t_n)$. The initial value $\boldsymbol{X}_0$ may be stochastic or deterministic. Observations are made on each unit, modeled by an observable process $\{\boldsymbol{Y}_n = Y_{1:U,n}, n \in 1\!:\!N\}$ which takes values in a product space $\mathbb{Y}^U$. Observations are modeled as being conditionally independent given the latent process. The conditional independence of measurements applies over both time and the unit structure, so the collection $\{Y_{u,n}, u \in 1\!:\!U, n \in 1 : N\}$ is conditionally independent given $\{X_{u,n}, u \in 1 : U, n \in 1 : N\}$. The unit structure for the observation process is not necessary for all that follows (see Appendix 2.C). We suppose the existence of a joint density $f_{\boldsymbol{X}_{0:N},\boldsymbol{Y}_{1:N}}$ of $X_{1:U,1:N}$ and $Y_{1:U,1:N}$ with respect to some appropriate measure, following a notational convention that the subscripts of f denote the joint or conditional density under consideration. The data are $y^*_{u,n}$ for unit u at time n. This model is a special case of a partially observed Markov process (POMP, Bretó et al., 2009), also known as a state space model or hidden Markov model. The additional unit structure, not generally required for a POMP, is appropriate for modeling interactions between units characterized by a spatial location, and so we call the model a SpatPOMP. In the following, we use the ordering on the set of observations corresponding to their unit labels $1\!:\!U$ to define the set of observations preceding unit u at time n as

$$A_{u,n} = \{(\tilde{u}, \tilde{n}) : 1 \leq \tilde{n} < n \text{ or } (\tilde{n} = n \text{ and } \tilde{u} < u)\}. \tag{2.1}$$

The ordering of the spatial locations in (2.1) might seem artificial, and indeed densities such as $f_{X_{u,n}|X_{A_{u,n}}}$ will frequently be hard to compute or simulate from. The bagged filter algorithms we study do not evaluate or simulate such transition densities but only compute the measurement model on neighborhoods, unlike the filter of Beskos et al. (2017) built on the same factorization. We may suppose that sufficiently distant units approach independence, in which case we say the system is *weakly coupled.* To represent weak coupling, we suppose there is a neighborhood $B_{u,n} \subset A_{u,n}$ such that the latent process on $A_{u,n} \setminus B_{u,n}$ is approximately conditionally independent of $X_{u,n}$ given data on $B_{u,n}$.

Our primary interest is estimation of the log likelihood for the data given the model, $\ell = \log f_{\boldsymbol{Y}_{1:N}}(\boldsymbol{y}^*_{1:N})$, which is of fundamental importance in both Bayesian and non-Bayesian statistical inference. A general filtering problem is to evaluate $\mathbb{E}\big[h(X_{u,n}) \,|\, Y_{A_{u,n}} = y^*_{A_{u,n}}\big]$ for some function $h : \mathbb{X} \to \mathbb{R}$. Taking $h(x) = f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, x)$ gives a filtering representation of the likelihood evaluation problem. Further discussion on bagged filtering for other filtering problems is given in Appendix 2.E. For likelihood-based inference, maximization plays an important role in point estimation, confidence interval construction, hypothesis testing and model selection. An extension of bagged filtering to likelihood maximization is demonstrated in Sec. 4.4.

Pseudocode for a UBF algorithm for likelihood evaluation is given below. The plug-and-play property is evident because UBF requires as input a simulator for the latent coupled dynamic process but not an evaluator of transition probabilities. The pseudocode for UBF adopts a convention that implicit loops are carried out over all free indices, meaning indices with values that are not explicitly specified. For example, the construction of $w^P_{u,n,i}$ in UBF has an implicit loop over u, n and i. However, the summation constructing $\ell^{\mathrm{MC}}_{u,n}$ does not have an implicit loop over i since the summation index i is specified explicitly and so is not a free index.

UBF. Unadapted bagged filter.

```
input:
    simulator for $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)$ and $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{x}_{n-1})$
    evaluator for $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \,|\, x_{u,n})$
    number of replicates, $\mathcal{I}$
    neighborhood structure, $B_{u,n}$
    data, $\boldsymbol{y}^*_n$
implicit loops:
    $u$ in $1\!:\!U$,  $n$ in $1\!:\!N$,  $i$ in $1\!:\!\mathcal{I}$
algorithm:
    simulate $\boldsymbol{X}_{0:N,i} \sim f_{\boldsymbol{X}_{0:N}}(\boldsymbol{x}_{0:N})$
    measurement weights, $w^M_{u,n,i} = f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, X_{u,n,i})$
    prediction weights, $w^P_{u,n,i} = \prod_{(\tilde{u},\tilde{n}) \in B_{u,n}} w^M_{\tilde{u},\tilde{n},i}$
    $\ell^{\mathrm{MC}}_{u,n} = \log\left(\sum_{i=1}^{\mathcal{I}} w^M_{u,n,i} w^P_{u,n,i}\right) - \log\left(\sum_{i=1}^{\mathcal{I}} w^P_{u,n,i}\right)$
output:
    log likelihood estimate, $\ell^{\,\mathrm{MC}} = \sum_{n=1}^{N} \sum_{u=1}^{U} \ell^{\mathrm{MC}}_{u,n}$
```

2.2.2 UBF theory

For each pair (U, N) we suppose there is a dataset $\boldsymbol{y}^*_{1:N}$ and a model $f_{\boldsymbol{X}_{0:N},\boldsymbol{Y}_{1:N}}$. We are interested in results that hold for all (U, N), but we do not require that the models for $(U_1, N_1) \neq (U_2, N_2)$ are related, except through satisfying regularity conditions. The following conditions impose requirements that distant regions of space-time behave similarly and have only weak dependence. The conditions are written non-asymptotically in terms of constants ϵ_{A1}, ϵ_{A4} and Q which are used to bound the asymptotic bias and variance in Theorem 1. Stronger bounds are obtained when the conditions hold for small ϵ_{A1}, ϵ_{A4} and Q.

Assumption A1. *There is an $\epsilon_{\mathrm{A1}} > 0$, independent of U and N, and a collection of neighborhoods $\{B_{u,n} \subset A_{u,n}, u \in 1\!:\!U, n \in 1\!:\!N\}$ such that, for all u and n, any bounded real-valued function $|h(x)| \leq 1$, and any value of $x_{B^c_{u,n}}$,*

$$\bigg| \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}, X_{B^c_{u,n}}}(x_{u,n} \,|\, y^*_{B_{u,n}}, x_{B^c_{u,n}})\, dx_{u,n} \\ - \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}}(x_{u,n} \,|\, y^*_{B_{u,n}})\, dx_{u,n} \bigg| \;<\; \epsilon_{\mathrm{A1}}. \tag{2.2}$$

Assumption A2. *For the collection of neighborhoods in Assumption A1, with $B^+_{u,n} = B_{u,n} \cup (u,n)$, there is a constant b, depending on ϵ_{A1} but not on U and N, such that*

$$\sup_{u \in 1:U,\, n \in 1:N} \big|B^+_{u,n}\big| \leq b.$$

Assumption A3. *There is a constant Q, independent of U and N, such that, for all u and n,*

$$Q^{-1} < f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, x_{u,n}) < Q$$

Assumption A4. *There exists $\epsilon_{\mathrm{A4}} > 0$, independent of U and N, such that the following holds.*

For each u, n, a set $C_{u,n} \subset (1\!:\!U) \times (0\!:\!N)$ exists such that $(\tilde{u}, \tilde{n}) \notin C_{u,n}$ implies $B^+_{u,n} \cap B^+_{\tilde{u},\tilde{n}} = \emptyset$ and

$$|f_{X_{B^+_{\tilde{u},\tilde{n}}}|X_{B^+_{u,n}}} - f_{X_{B^+_{\tilde{u},\tilde{n}}}}| < \epsilon_{\mathrm{A4}}\, f_{X_{B^+_{\tilde{u},\tilde{n}}}}$$

Further, there is a uniform bound $|C_{u,n}| \leq c$.

The two mixing conditions in Assumptions A1 and A4 are subtly different. Assumption A1 describes a conditional mixing property dependent on the data, whereas A4 asserts a form of unconditional mixing. Although both capture a similar concept of weak coupling, conditional and unconditional mixing properties do not readily imply one another. Assumption A3 is a compactness condition of a type that has proved useful in the theory of particle filters despite the rarity of its holding exactly. Theorem 1 shows that these conditions let UBF compute the likelihood with a Monte Carlo variance of order $UN\mathcal{I}^{-1}$ with a bias of order $UN\epsilon$.

Theorem 1. *Let ℓ^{MC} denote the Monte Carlo likelihood approximation constructed by UBF. Consider a limit with a growing number of bootstrap replicates, $\mathcal{I} \to \infty$, and suppose assumptions A1, A2 and A3. There are quantities $\epsilon(U, N)$ and $V(U, N)$, with bounds $|\epsilon| < \epsilon_{\mathrm{A1}} Q^2$ and $V < Q^{4b} U^2 N^2$, such that*

$$\mathcal{I}^{1/2}\big[\ell^{MC} - \ell - \epsilon UN\big] \xrightarrow[\mathcal{I}\to\infty]{d} \mathcal{N}[0, V], \tag{2.3}$$

where $\xrightarrow[\mathcal{I}\to\infty]{d}$ denotes convergence in distribution and $\mathcal{N}[\mu, \Sigma]$ is the normal distribution with mean μ and variance Σ. If additionally Assumption A4 holds, we obtain an improved variance bound

$$V < Q^{4b}\, UN\big(c + \epsilon_{\mathrm{A4}}\,(UN - c)\big). \tag{2.4}$$

Proof. A complete proof is given in Appendix 2.A. Briefly, the assumptions imply a multivariate central limit theorem for $\{\ell^{\mathrm{MC}}_{u,n}, (u,n) \in 1\!:\!U \times 1\!:\!N\}$ as $\mathcal{I} \to \infty$. The limiting variances and covariances are uniformly bounded, using Assumptions A2 and A3. Assumption A1 provides a uniform bound on the discrepancy between $\ell_{u,n}$ and mean of the Gaussian limit. This is enough to derive (2.3). Assumption A4 gives a stronger bound on covariances between sufficiently distant units, leading to (2.4). $\square$

Theorem 1 does not guarantee uniformity over U and N of the rate of convergence as $\mathcal{I} \to \infty$. However, it does guarantee that the polynomial bounds in (2.3) and (2.4) hold for sufficiently large $\mathcal{I}$. The COD is characterized by exponential bounds, and so Theorem 1 shows a specific sense in which UBF can avoid COD. Uniformity of the central limit convergence in Theorem 1 may be expected to hold via a Berry-Esseen theorem, but extension of existing Berry-Esseen results for dependent processes (Bentkus et al., 1997; Jirak, 2016) is beyond the scope of this article.

The approximation error for UBF can be divided into two sources: a localization bias due to conditioning on a finite neighborhood, and Monte Carlo error. The localization bias does not disappear in the limit as Monte Carlo effort increases. It does become small as the conditioning neighborhood increases, but the Monte Carlo effort grows exponentially in the size of this neighborhood. The class of problems for which these algorithms are useful are ones where a relatively small neighborhood is adequate. Although the filtering inference is carried out using localization, the simulation of the process is carried out globally which avoids the introduction of additional boundary effects and ensures that the simulations comply with any constraint properties of the global process simulator.

2.3 Adaptation and intermediate resampling

2.3.1 Concept

Theorem 1 shows that UBF can beat COD. However, UBF can perform poorly on long time series unless weak temporal dependence allows simulated sample paths to remain relevant over the course of a long time series. Later, in Sec. 4.2, we will find that UBF performs well on an epidemiological model but less well on a geophysical model. It may in practice be necessary to select simulations consistent with the data, much as standard PF algorithms do. We look for approaches that build on the basic insight of UBF while having superior practical performance.

Whereas the full global filtering problem of drawing from $f_{\boldsymbol{X}_n|\boldsymbol{Y}_{1:n}}$ may be intractable via importance sampling methods, a version of this problem localized in space and time may nevertheless be feasible. The conditional density, $f_{\boldsymbol{X}_n|\boldsymbol{Y}_n,\boldsymbol{X}_{n-1}}$, is called the *adapted density*, and simulating from this density is called *adapted simulation*. For models where $\boldsymbol{X}_{n-1}$ is highly informative about $\boldsymbol{X}_n$, importance sampling for adapted simulation may be much easier than the full filter calculation. The following adapted bagged filter (ABF) is constructed under a hypothesis that the adapted simulation problem is tractable, and it is applicable when the number of units is prohibitive for Monte Carlo sampling from the full filter distribution but not for sampling from the adapted distribution. In ABF, the adapted simulations are reweighted in a neighborhood of each unit and time point to construct a local approximation to the filtering problem which leads to an estimate of the likelihood. The pseudocode for ABF, below, reduces to UBF when using a single particle per repicate, $J = 1$.

ABF. Adapted bagged filter.

```
input: same as for UBF plus
    particles per replicate, $J$
implicit loops:
    $u$ in $1\!:\!U$,  $n$ in $1\!:\!N$,  $i$ in $1\!:\!\mathcal{I}$,  $j$ in $1\!:\!J$
algorithm:
    Initialize adapted simulation: $\boldsymbol{X}^{\mathrm{A}}_{0,i} \sim f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)$
    For $n$ in $1\!:\!N$
        Proposals: $\boldsymbol{X}^{\mathrm{P}}_{n,i,j} \sim f_{\boldsymbol{X}_n|X_{1:U,n-1}}\big(\boldsymbol{x}_n \,|\, \boldsymbol{X}^{\mathrm{A}}_{n-1,i}\big)$
        Measurement weights: $w^{M}_{u,n,i,j} = f_{Y_{u,n}|X_{u,n}}\big(y^{*}_{u,n} \,|\, X^{\mathrm{P}}_{u,n,i,j}\big)$
        Adapted resampling weights: $w^{\mathrm{A}}_{n,i,j} = \prod_{u=1}^{U} w^{M}_{u,n,i,j}$
        Resampling: $\mathbb{P}\big[r(i)=a\big] = w^{\mathrm{A}}_{n,i,a}\Big(\sum_{k=1}^{J} w^{\mathrm{A}}_{n,i,k}\Big)^{-1}$
        $\boldsymbol{X}^{\mathrm{A}}_{n,i} = \boldsymbol{X}^{\mathrm{P}}_{n,i,r(i)}$
        $w^{\mathrm{P}}_{u,n,i,j} = \prod_{\tilde{n}=1}^{n-1}\Big[\frac{1}{J}\sum_{k=1}^{J}\prod_{\tilde{u}:(\tilde{u},\tilde{n})\in B_{u,n}} w^{M}_{\tilde{u},\tilde{n},i,k}\Big]\prod_{\tilde{u}:(\tilde{u},n)\in B_{u,n}} w^{M}_{\tilde{u},n,i,j}$
    End for
output:
    $\ell^{\mathrm{MC}}_{u,n} = \log\left(\dfrac{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^{M}_{u,n,i,j} w^{P}_{u,n,i,j}}{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^{P}_{u,n,i,j}}\right)$
```

ABF remedies a weakness of UBF by making each boostrap filter adapted to the data. However, this benefit carries a cost, since adapted simulation is not immune from the curse of dimensionality. Therefore, we also consider an algorithm called ABF-IR which uses an intermediate resampling technique to carry out the adapted simulation. Intermediate resampling involves assessing the satisfactory progress of particles toward the subsequent observation at a collection of times between observations. This is well defined when the latent process has a continuous time representation,

$\{\boldsymbol{X}(t)\}$, with observation times $t_{1:N}$. We write S intermediate resampling times as

$$t_{n-1} = t_{n,0} < t_{n,1} < \cdots < t_{n,S} = t_n.$$

Carrying out an intermediate resampling procedure can have favorable scaling properties when S is proportional to U (Park and Ionides, 2020). In the case $S = 1$, ABF-IR reduces to ABF. Intermediate resampling was developed in the context of sequential Monte Carlo (Del Moral and Murray, 2015; Park and Ionides, 2020); however, the same theory and methodology can be applied to the simpler and easier problem of adapted simulation. ABF-IR employs a guide function to gauge the compatibility of each particle with future data. This is a generalization of the popular auxiliary particle filter (Pitt and Shepard, 1999). Only an ideal guide function fully addresses COD (Park and Ionides, 2020) and on nontrivial problems this is not available. However, practical guide functions can nevertheless improve performance.

The implementation in the ABF-IR pseudocode constructs the guide $g_{n,s,i,j}$ using a simulated moment method proposed by Park and Ionides (2020). The quantities $\boldsymbol{X}^G_{n,i,j}$, $V_{u,n,i}$, $\boldsymbol{\mu}^{\mathrm{IP}}_{n,s,i,j}$, $V^{\mathrm{meas}}_{u,n,s,i,j}$, $V^{\mathrm{proc}}_{u,n,s,i}$ and $\theta_{u,n,s,i,j}$ constructed in ABF-IR are used only to construct $g_{n,s,i,j}$. Heuristically, we use guide simulations to approximate the variance of the increment in each particle between time points, and we augment the measurement variance to account for both dynamic variability and measurement error. The guide function affects numerical performance of the algorithm but not its correctness: it enables a computationally convenient approximation to improve performance on the intractable target problem. Our guide function supposes the availability of a deterministic function approximating evolution of the mean of the latent process, written as

$$\boldsymbol{\mu}(\boldsymbol{x}, s, t) \approx \mathbb{E}\big[\boldsymbol{X}(t) \,|\, \boldsymbol{X}(s) = \boldsymbol{x}\big].$$

Further, the guide requires that the measurement model has known conditional mean and variance as a function of the model parameter vector θ, written as

$$\begin{aligned} h_{u,n}(x_{u,n}) &= \mathbb{E}\big[Y_{u,n} \,|\, X_{u,n} = x_{u,n}\big] \\ \overrightarrow{\mathrm{v}}_{u,n}(x_{u,n}, \theta) &= \mathrm{Var}\big(Y_{u,n} \,|\, X_{u,n} = x_{u,n}\,; \theta\big) \end{aligned}$$

Also required for ABF-IR is an inverse function $\overleftarrow{\mathrm{v}}_{u,n}$ such that

$$\overrightarrow{\mathrm{v}}_{u,n}\big(x_{u,n}, \overleftarrow{\mathrm{v}}_{u,n}(V, x_{u,n}, \theta)\big) = V.$$

ABF-IR. Adapted bagged filter with intermediate resampling.

```
input: same as for ABF plus
  number of intermediate timesteps, $S$
  measurement variance parameterizations, $\overleftarrow{\mathrm{v}}_{u,n}$ and $\overrightarrow{\mathrm{v}}_{u,n}$
  approximate process and observation mean functions, $\boldsymbol{\mu}$ and $h_{u,n}$
implicit loops:
  $u$ in $1\!:\!U$, $n$ in $1\!:\!N$, $i$ in $1\!:\!\mathcal{I}$, $j$ in $1\!:\!J$, $j$ in $1\!:\!J$
algorithm:
  Initialize adapted simulation: $\boldsymbol{X}^{\mathrm{A}}_{0,i} \sim f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)$
  For $n$ in $1\!:\!N$
    Guide simulations: $\boldsymbol{X}^{G}_{n,i,j} \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{X}^{\mathrm{A}}_{n-1,i})$
    Guide sample variance: $V_{u,n,i} = \mathrm{Var}\{h_{u,n}(X^{G}_{u,n,i,j}), j \text{ in } 1\!:\!J\}$
    $g^{\mathrm{R}}_{n,0,i,j} = 1$ and $\boldsymbol{X}^{\mathrm{IR}}_{n,0,i,j} = \boldsymbol{X}^{\mathrm{A}}_{n-1,i}$
    For $s$ in $1\!:\!S$
      Intermediate proposals: $\boldsymbol{X}^{\mathrm{IP}}_{n,s,i,j} \sim f_{\boldsymbol{X}_{n,s}|\boldsymbol{X}_{n,s-1}}(\,\cdot\,|\boldsymbol{X}^{\mathrm{IR}}_{n,s-1,i,j})$
      $\boldsymbol{\mu}^{\mathrm{IP}}_{n,s,i,j} = \boldsymbol{\mu}(\boldsymbol{X}^{\mathrm{IP}}_{n,s,i,j}, t_{n,s}, t_n)$
      $V^{\mathrm{meas}}_{u,n,s,i,j} = \overrightarrow{\mathrm{v}}_u(\theta, \mu^{\mathrm{IP}}_{u,n,s,i,j})$
      $V^{\mathrm{proc}}_{u,n,s,i} = V_{u,n,i}\,(t_n - t_{n,s}) \big/ (t_n - t_{n,0})$
      $\theta_{u,n,s,i,j} = \overleftarrow{\mathrm{v}}_u(V^{\mathrm{meas}}_{u,n,s,i,j} + V^{\mathrm{proc}}_{u,n,s,i},\, \mu^{\mathrm{IP}}_{u,n,s,i,j})$
      $g_{n,s,i,j} = \prod_{u=1}^{U} f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, \mu^{\mathrm{IP}}_{u,n,s,i,j}\,; \theta_{u,n,s,i,j})$
      Guide weights: $w^{G}_{n,s,i,j} = g_{n,s,i,j} / g^{\mathrm{R}}_{n,s-1,i,j}$
      Resampling: $\mathbb{P}[r(i,j)=a] = w^{G}_{n,s,i,a}\Big(\sum_{k=1}^{J} w^{G}_{n,s,i,k}\Big)^{-1}$
      $\boldsymbol{X}^{\mathrm{IR}}_{n,s,i,j} = \boldsymbol{X}^{\mathrm{IP}}_{n,s,i,r(i,j)}$ and $g^{\mathrm{R}}_{n,s,i,j} = g_{n,s,i,r(i,j)}$
    End For
    Set $\boldsymbol{X}^{\mathrm{A}}_{n,i} = \boldsymbol{X}^{\mathrm{IR}}_{n,S,i,1}$
    Measurement weights: $w^{M}_{u,n,i,j} = f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, X^{G}_{u,n,i,j})$
    $w^{\mathrm{P}}_{u,n,i,j} = \prod_{\tilde{n}=1}^{n-1}\Big[\frac{1}{J}\sum_{a=1}^{J}\prod_{\tilde{u}:(\tilde{u},\tilde{n})\in B_{u,n}} w^{M}_{\tilde{u},\tilde{n},i,a}\Big] \prod_{\tilde{u}:(\tilde{u},n)\in B_{u,n}} w^{M}_{\tilde{u},n,i,j}$
  End for
output:
  $\ell^{\mathrm{MC}}_{u,n} = \log\left(\dfrac{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^{M}_{u,n,i,j} w^{P}_{u,n,i,j}}{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^{P}_{u,n,i,j}}\right)$
```

This guide function is applicable to spatiotemporal versions of a broad range of population and compartment models used to model dynamic systems in ecology, epidemiology, and elsewhere. Other guide functions could be developed and inserted into the ABF-IR algorithm, including other constructions considered by Park and Ionides (2020).

One might wonder why it is appropriate to keep many particle representations at intermediate timesteps while resampling down to a single representative at each observation time. Part of the answer is that adaptive simulation can fail when one resamples down to a single particle too often (Appendix 2.D).

2.3.2 ABF-IR theory

We start by considering a deterministic limit for infinite Monte Carlo effort and explaining why the ABF and ABF-IR algorithms approximately target the likelihood function, subject to suitable mixing behavior. Subsequently, we consider the scaling properties as Monte Carlo effort increases. We adopt a convention that densities involving $Y_{u,n}$ are implicitly evaluated at the data, $y^*_{u,n}$, and densities involving $X_{u,n}$ are implicitly evaluated at $x_{u,n}$ unless otherwise specified. We write $A^+_{u,n} = A_{u,n} \cup (u,n)$, matching the defintion $B^+_{u,n} = B_{u,n} \cup (u,n)$. The essential ingredient in all the algorithms is a localization of the likelihood, which may be factorized sequentially as

$$f_{Y_{1:U,1:N}} = \prod_{n=1}^{N} \prod_{u=1}^{U} f_{Y_{u,n}|Y_{A_{u,n}}} = \prod_{n=1}^{N} \prod_{u=1}^{U} \frac{f_{Y_{A^+_{u,n}}}}{f_{Y_{A_{u,n}}}}.$$

In particular, the approximations assume that the full history $A_{u,n}$ can be well approximated by a neighborhood $B_{u,n} \subset A_{u,n}$. UBF approximates $f_{Y_{u,n}|Y_{A_{u,n}}}$ by

$$f_{Y_{u,n}|Y_{B_{u,n}}} = \frac{f_{Y_{B^+_{u,n}}}}{f_{Y_{B_{u,n}}}} = \frac{\int f_{Y_{B^+_{u,n}}|X_{B^+_{u,n}}} f_{X_{B^+_{u,n}}} \, dx_{B^+_{u,n}}}{\int f_{Y_{B_{u,n}}|X_{B_{u,n}}} f_{X_{B_{u,n}}} \, dx_{B_{u,n}}}.$$

For $B \subset 1\!:\!U \times 1\!:\!N$, define $B^{[m]} = B \cap \big(1\!:\!U \times \{m\}\big)$. ABF and ABF-IR build on the following identity,

$$f_{Y_{A_{u,n}}} = \int f_{\boldsymbol{X}_0} \left[\prod_{m=1}^{n} f_{\boldsymbol{X}_m|\boldsymbol{X}_{m-1},\boldsymbol{Y_m}} f_{Y_{A^{[m]}_{u,n}}|\boldsymbol{X}_{m-1}} \right] d\boldsymbol{x}_{0:n},$$

where $f_{\boldsymbol{X}_m|\boldsymbol{X}_{m-1},\boldsymbol{Y}_m}$ is called the adapted transition density. The adapted process (i.e., a stochastic process following the adapted transition density) can be interpreted as a one-step greedy procedure using the data to guide the latent process. Let $g_{\boldsymbol{X}_{0:N},\boldsymbol{X}^P_{1:N}}(\boldsymbol{x}_{0:N}, \boldsymbol{x}^P_{1:N})$ be the joint density of the adapted process and the proposal process,

$$\begin{aligned} g_{\boldsymbol{X}_{0:N},\boldsymbol{X}^P_{1:N}}(\boldsymbol{x}_{0:N}, \boldsymbol{x}^P_{1:N}) &= f_{\boldsymbol{X}_0}(\boldsymbol{x}_0) \times \\ \prod_{n=1}^{N} f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1},\boldsymbol{Y}_n}(\boldsymbol{x}_n \,|\, \boldsymbol{x}_{n-1}, \boldsymbol{y}^*_n) \; & f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}^P_n \,|\, \boldsymbol{x}_{n-1}). \end{aligned} \tag{2.5}$$

Using the convention that an empty density $f_{Y_\emptyset}$ evaluates to 1, we define

$$\gamma_B = \prod_{m=1}^{N} f_{Y_{B^{[m]}}|\boldsymbol{X}_{m-1}}(y^*_{B^{[m]}} \,|\, \boldsymbol{X}_{m-1}).$$

Denoting $\mathbb{E}_g$ for expectation for $(\boldsymbol{X}_{0:N}, \boldsymbol{X}^P_{1:N})$ having density $g_{\boldsymbol{X}_{0:N},\boldsymbol{X}^P_{1:N}}$, we have

$$f_{Y_{u,n}|Y_{A_{u,n}}} = \frac{\mathbb{E}_g\big[\gamma_{A^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{A_{u,n}}\big]}.$$

Estimating this ratio by Monte Carlo sampling from g is problematic due to the growing size of $A_{u,n}$. Thus, ABF and ABF-IR make a localized approximation,

$$\frac{\mathbb{E}_g\left[\gamma_{A^+_{u,n}}\right]}{\mathbb{E}_g\left[\gamma_{A_{u,n}}\right]} \approx \frac{\mathbb{E}_g\left[\gamma_{B^+_{u,n}}\right]}{\mathbb{E}_g\left[\gamma_{B_{u,n}}\right]}. \tag{2.6}$$

The conditional log likelihood estimate $\ell^{\mathrm{MC}}_{u,n}$ in ABF and ABF-IR come from replacing the expectations on the right hand side of (2.6) with averages over Monte Carlo replicates of simulations from the adapted process. To see that we expect the approximation in (2.6) to hold when dependence decays across spatiotemporal distance, we can write

$$\begin{aligned}
\gamma_{A_{u,n}} &= \gamma_{B_{u,n}}\,\gamma_{B^c_{u,n}} \\
\gamma_{A^+_{u,n}} &= \gamma_{B^+_{u,n}}\,\gamma_{B^c_{u,n}},
\end{aligned}$$

where $B^c_{u,n}$ is the complement of $B_{u,n}$ in $A_{u,n}$. Under our assumptions, the term corresponding to $\gamma_{B^c_{u,n}}$ approximately cancels in the numerator and denominator of the right hand side of (2.6).

Since ABF is ABF-IR with $S = 1$, we focus attention on ABF-IR. At a conceptual level, the localized likelihood estimate in ABF-IR has the same structure as its UBF counterpart. However, ABF-IR additionally requires the capability to satisfactorily implement adapted simulation. Adapted simulation is a local calculation, making it an easier task than the global operation of filtering. Nevertheless, adapted simulation via importance sampling is vulnerable to COD for sufficiently large values of U. For a continuous time model, the use of $S > 1$ is motivated by a result that guided intermediate resampling can reduce, or even remove, the COD in the context of a particle filtering algorithm (Park and Ionides, 2020). Assumptions B1–B4 below are analogous to A1–A4 and are non-asymptotic assumptions involving $\epsilon_{\text{B1}} > 0$, $\epsilon_{\text{B4}} > 0$ and $Q > 1$ which are required to hold uniformly over space and time. Assumptions B5–B7 control the Monte Carlo error arising from adapted simulation. B5 is a stability property which asserts that the effect of the latent process on the future of the adapted process decays over time. Assumption B6 is a non-asymptotic bound on Monte Carlo error for a single step of adapted simulation. The scaling of the constant $\mathcal{C}_0$ with U, N and S in Assumption B6 has been studied by Park and Ionides (2020), where it was established that setting $S = U$ can lead to $\mathcal{C}_0$ being constant, when using an ideal guide function, or slowly growing with U otherwise. The $\epsilon_{\text{B6}}^{-3}$ error rate in Assumption B6 follows from balancing the two sources of error defined in the statement of Theorem 2 of Park and Ionides (2020). Assumption B7 can be guaranteed by the construction of the algorithm, if independently generated Monte Carlo random variables are used for building the guide function and the one-step prediction particles. The asymptotic limit in Theorem 2 arises as the number of replicates increases.

Assumption B1. *There is an $\epsilon_{\text{B1}} > 0$, independent of U and N, and a collection of neighborhoods $\{B_{u,n} \subset A_{u,n}, u \in 1\!:\!U, n \in 1\!:\!N\}$ such that the following holds for all u and n, and any bounded real-valued function $|h(x)| \leq 1$: if we write $A = A_{u,n}$, $B = B_{u,n}$, $f_A(x_A) = f_{Y_A|X_A}(y_A^*|x_A)$, and $f_B(x_B) = f_{Y_B|X_B}(y_B^*|x_B)$,*

$$\left| \int h(x) \left\{ \frac{\mathbb{E}_g[f_A(X_A^P) f_{X_{u,n}|X_{A[n]},\boldsymbol{X}_{n-1}}(x|X_{A[n]}^P, \boldsymbol{X}_{n-1})]}{\mathbb{E}_g[f_A(X_A^P)]} - \frac{\mathbb{E}_g[f_B(X_B^P) f_{X_{u,n}|X_{B[n]},\boldsymbol{X}_{n-1}}(x|X_{B[n]}^P, \boldsymbol{X}_{n-1})]}{\mathbb{E}_g[f_B(X_B^P)]} \right\} dx \right| < \epsilon_{\text{B1}}.$$

Assumption B2. *The bound $\sup_{u\in 1:U, n\in 1:N} \left|B_{u,n}^+\right| \leq b$ in Assumption A2 applies for the neighborhoods defined in Assumption B1. This also implies there is a finite maximum temporal depth for* *the collection of neighborhoods, defined as*

$$d_{\max} = \sup_{(u,n)} \sup_{(\tilde{u},\tilde{n})\in B_{u,n}} |n-\tilde{n}|.$$

Assumption B3. *Identically to Assumption A3,* $Q^{-1} < f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, x_{u,n}) < Q$.

Assumption B4. *We use subscripts of* g *to denote marginal and conditional densities derived from (2.20). Suppose there is an* ϵ_{B4}*, independent of* U *and* N*, such that the following holds. For each* u *and* n*, a set* $C_{u,n} \subset (1\!:\!U)\times(0\!:\!N)$ *exists such that* $(\tilde{u},\tilde{n}) \notin C_{u,n}$ *implies* $B^+_{u,n} \cap B^+_{\tilde{u},\tilde{n}} = \emptyset$ *and*

$$|g_{X^P_{B_{\tilde{u},\tilde{n}}\cup B_{u,n}}} - g_{X^P_{B_{\tilde{u},\tilde{n}}}}\, g_{X^P_{B_{u,n}}}| < (1/2)\,\epsilon_{\mathrm{B4}}\; g_{X^P_{B_{\tilde{u},\tilde{n}}\cup B_{u,n}}}$$

$$\begin{aligned}|g_{X^P_{B_{\tilde{u},\tilde{n}}}|\boldsymbol{X}_{0:N}}\, g_{X^P_{B_{u,n}}|\boldsymbol{X}_{0:N}} - g_{X^P_{B_{\tilde{u},\tilde{n}}\cup B_{u,n}}|\boldsymbol{X}_{0:N}}| \\ < (1/2)\,\epsilon_{\mathrm{B4}}\; g_{X^P_{B_{\tilde{u},\tilde{n}}\cup B_{u,n}}|\boldsymbol{X}_{0:N}}\end{aligned}$$

Further, there is a uniform bound $|C_{u,n}| \le c$.

Assumption B5. *There is a constant* K*, independent of* U *and* N*, such that, for any* $0 \le d \le d_{\max}$*, any* $n \ge K+d$*, and any set* $D \subset (1\!:\!U)\times(n\!:\!n-d)$*,*

$$\begin{aligned}|g_{X_D|\boldsymbol{X}_{n-d-K}}(x_D \,|\, \boldsymbol{x}^{(1)}_{n-d-K}) - g_{X_D|\boldsymbol{X}_{n-d-K}}(x_D \,|\, \boldsymbol{x}^{(2)}_{n-d-K})| \\ < \epsilon_{\mathrm{B5}}\; g_{X_D|\boldsymbol{X}_{n-d-K}}(x_D \,|\, \boldsymbol{x}^{(1)}_{n-d-K})\end{aligned}$$

holds for all $\boldsymbol{x}^{(1)}_{n-d-K}$, $\boldsymbol{x}^{(2)}_{n-d-K}$, *and* x_D.

Assumption B6. *Let h be a bounded function with $|h(x)| \leq 1$. Let $\boldsymbol{X}^{\mathrm{IR}}_{n,S,j,i}$ be the Monte Carlo quantity constructed in ABF-IR, conditional on $\boldsymbol{X}^{\mathrm{A}}_{n-1,S,i} = \boldsymbol{x}^{\mathrm{A}}_{n-1,S,i}$. There is a constant $\mathcal{C}_0(U,N,S)$ such that, for all $\epsilon_{\mathrm{B6}} > 0$ and $\boldsymbol{x}^{\mathrm{A}}_{n-1,S,i}$, whenever the number of particles satisfies $J > \mathcal{C}_0(U,N,S)/\epsilon^3_{\mathrm{B6}}$,*

$$\left| \mathbb{E}\Big[\frac{1}{J}\sum_{j=1}^{J} h(\boldsymbol{X}^{\mathrm{IR}}_{n,S,j,i})\Big] - \mathbb{E}_g\big[h(\boldsymbol{X}_n) \,|\, \boldsymbol{X}_{n-1} = \boldsymbol{x}^{\mathrm{A}}_{n-1,S,i}\big] \right| < \epsilon_{\mathrm{B6}}.$$

Assumption B7. *For $1 \leq n \leq N$, the Monte Carlo random variable $X^A_{n,i}$ is independent of $w^M_{u,n,i,j}$ conditional on $X^A_{n-1,i}$.*

Theorem 2. *Let $\ell^{\,MC}$ denote the Monte Carlo likelihood approximation constructed by ABF-IR, or by ABF since this is the special case of ABF-IR with $S = 1$. Consider a limit with a growing number of bootstrap replicates, $\mathcal{I} \to \infty$, and suppose assumptions B1, B2, B3, B5, B6 and B7. Suppose the number of particles J exceeds the requirement for B6. There are quantities $\epsilon(U,N)$ and $V(U,N)$ with $|\epsilon| < Q^2\epsilon_{\mathrm{B1}} + 2Q^{2b}\big(\epsilon_{\mathrm{B5}} + (K+d_{\max})\epsilon_{\mathrm{B6}}\big)$ and $V < Q^{4b}U^2N^2$ such that*

$$\mathcal{I}^{1/2}\big[\ell^{\,MC} - \ell - \epsilon U N\big] \xrightarrow[\mathcal{I}\to\infty]{d} \mathcal{N}[0,V]. \tag{2.7}$$

If additionally Assumption B4 holds, we obtain an improved rate of

$$V < Q^{4b}NU\{c + \big(\epsilon_{\mathrm{B4}} + 3\epsilon_{\mathrm{B5}} + 4(K+d_{\max})\,\epsilon_{\mathrm{B6}}\big)(NU - c)\} \tag{2.8}$$

Proof. A full proof is provided in Sec. Appendix 2.B. The extra work to prove Theorem 2 beyond the argument for Theorem 1 is to bound the error arising from the importance sampling approximation to a draw from the adapted transition density. This bound is constructed using Assumptions B5, B6 and B7. The remainder of the proof follows the same approach as Theorem 1, with the adapted process replacing the unconditional latent process. □

The theoretical results foreshadow our empirical observations (Secs. 3.5 and 4.2) that the relative performance of UBF, ABF and ABF-IR is situation-dependent. Assumption A4 is a mixing assumption for the unconditional latent process, whereas Assumption B4 replaces this with a mixing assumption for the adapted process conditional on the data. For a non-stationary process, Assumption A4 may fail to hold uniformly in U whereas the adapted process may provide stable tracking of the latent process (Sec. Appendix 2.D). When Assumption A4 holds, UBF can benefit from not requiring Assumptions B5, B6 and B7. Adapted simulation is an easier problem than filtering, but nevertheless can become difficult in high dimensions, with the consequence that Assumption B6 could require large $\mathcal{C}_0$. The tradeoff between ABF and ABF-IR depends on the effectiveness of the guide function for the problem at hand. Intermediate resampling and guide function calculation require additional computational resources, which will necessitate smaller values of $\mathcal{I}$ and J. In some situations, the improved scaling properties of ABF-IR compared to ABF, corresponding to a lower value of $\mathcal{C}_0$, will outweigh this cost.

2.4 Examples

We compare the performance of the three bagged filters (UBF, ABF and ABF-IR) against each other and against alternative plug-and-play approaches. The plug-and-play property facilitates numerical implementation for general classes of models, and all the algorithms and models under consideration are implemented in the `spatPomp` R package (Asfaw et al., 2021b). Ensemble Kalman filter (EnKF) methods propagate the ensemble members by simulation from the dynamic model and then update the ensemble to assimilate observations using a Gaussian-inspired rule (Evensen and van Leeuwen, 1996; Lei et al., 2010). The block particle filter (BPF, Rebeschini and van Handel, 2015; Ng et al., 2002) partitions the latent space and combines independently drawn components from each partition. BPF overcomes COD under weak coupling assumptions (Rebeschini and van Handel, 2015). Unlike these two methods, our bagged filters modify particles only according to the latent dynamics. Thus, our methods satisfy any conservation laws, continuity or smoothness that arise when simulating from the dynamic model. We also compare with a guided intermediate resampling filter (GIRF, Park and Ionides, 2020), one of many variants of the particle filter designed to scale to larger numbers of units than are possible with a basic particle filter.

First, in Sec. 2.4.1, we consider a spatiotemporal Gaussian process for which the exact likelihood is available via a Kalman filter. We see in Fig. 2.1 that ABF-IR can have a considerable advantages over UBF and ABF for problems with an intermediate level of coupling. Then, in Sec. 2.4.2, we compare performance on the Lorenz-96 model, a highly coupled system used to test inference methods for geophysical applications.

2.4.1 Correlated Brownian motion

Suppose $\boldsymbol{X}(t) = \Omega \boldsymbol{W}(t)$ where $\boldsymbol{W}(t) = W_{1:U}(t)$ comprises U independent standard Brownian motions, and $\Omega_{u,\tilde{u}} = \rho^{d(u,\tilde{u})}$ with $d(u,\tilde{u})$ being the circle distance,

$$d(u,\tilde{u}) = \min\big(|u-\tilde{u}|, |u-\tilde{u}+U|, |u-\tilde{u}-U|\big).$$

Set $t_n = n$ for $n = 0, 1, \dots, N$ with initial value $\boldsymbol{X}(0) = \boldsymbol{0}$ and suppose measurement errors are independent and normally distributed, $Y_{u,n} = X_{u,n} + \eta_{u,n}$ with $\eta_{u,n} \sim \mathcal{N}(0, \tau^2)$. The parameter ρ determines the strength of the spatial coupling.

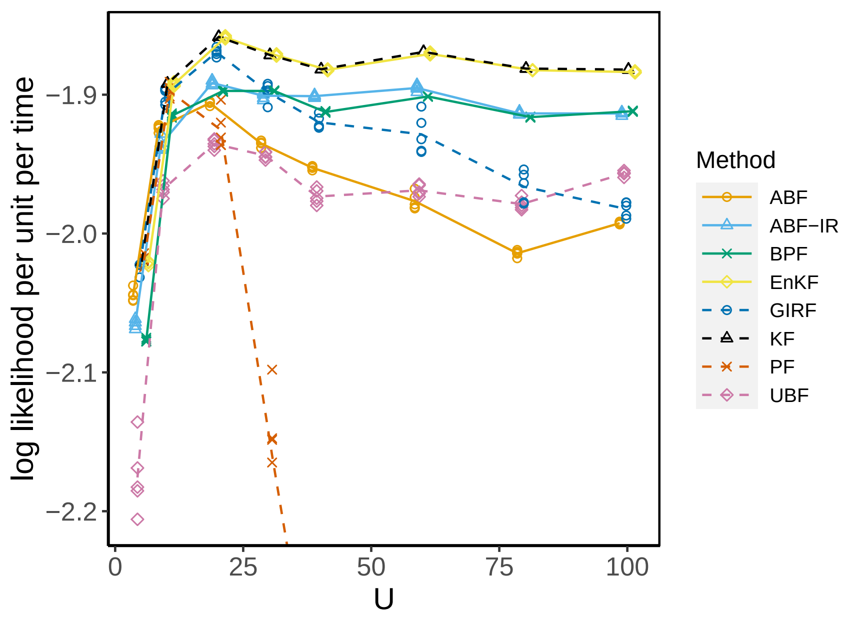


Figure 2.1 – Log likelihood estimates for correlated Brownian motions of various dimensions. UBF, ABF and ABF-IR are compared with a guided intermediate resampling filter (GIRF), standard particle filter (PF), block particle filter (BPF) and ensemble Kalman filter (EnKF). The exact likelihood was computed via a Kalman filter (KF).

Fig. 2.1 shows how the bagged filters scale on this Gaussian model, compared to a standard particle filter (PF), a guided intermediate resampling filter (GIRF), a block particle filter (BPF), and an ensemble Kalman filter. For our numerical results, we use $\tau = 1$, $\rho = 0.4$ and $N = 50$. In this case, the exact likelihood is computable via the Kalman filter (KF). Since EnKF is based on a Gaussian approximation, it is also exact in this case, up to a small Monte Carlo error. The GIRF framework encompasses lookahead particle filter techniques, such as the auxiliary particle filter (Pitt and Shepard, 1999), and intermediate resampling techniques (Del Moral et al., 2017). GIRF methods combining these techniques were found to perform better than either of these component techniques alone (Park and Ionides, 2020). Thus, GIRF here represents a state-of-the-art auxiliary particle filter that targets the complete joint filter density for all units. We use the general-purpose, plug-and-play implementation of GIRF provided by the `spatPomp` R package (Asfaw et al., 2021a);

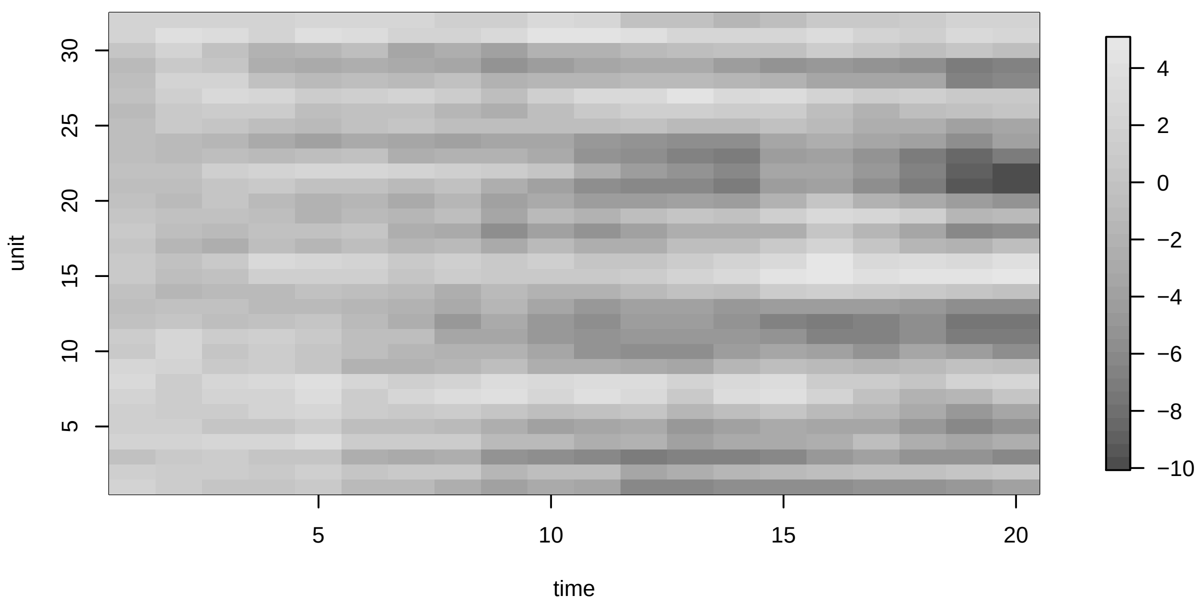


Figure 2.2 – Correlated Brownian motion simulation used in the analysis

for a Gaussian model, one can calculate an ideal guide function for GIRF but that was not used. PF works well for small values of U in Fig. 2.1 and rapidly starts struggling as U increases. GIRF behaves comparably to PF for small U but its performance is maintained for larger U. ABF and ABF-IR have some efficiency loss, for small U, relative to PF and GIRF due to the localization involved in the filter weighting, but for large U this cost is paid back by the benefit of the reduced Monte Carlo variability. UBF has a larger efficiency loss for small U, but its favorable scaling properties lead it to overtake ABF for larger U. BPF shows stable scaling and modest efficiency loss. This linear Gaussian SpatPOMP model provides a simple scenario to demonstrate scaling behavior. For filters that cannot take direct advantage of the Gaussian property of the model, we see that there is a tradeoff between efficiency at low U and scalability. This is unavoidable, since there is no known algorithm that is simultaneously fully efficient (up to Monte Carlo error), scalable, and applicable to general SpatPOMP models.

To help visualize the correlated Brownian motion model, Fig. 2.2 shows one of the simulations used for the results in Fig. 2.1 above. Table 2.1 gives the algorithmic settings used for the filters and corresponding computational resource requirements. Broadly speaking, $\mathcal{I}J$ for ABF and ABF-IR should be compared with $\mathcal{I}$ for UBF, J for PF, and JG for GIRF. The computational effort allocated to each algorithm in Table 2.1 is given in core minutes. UBF, ABF and ABF-IR parallelize readily, which is less true for PF and GIRF. Therefore, the UBF, ABF and ABF-IR implementations run on all available cores (36 for this experiment) whereas the PF and GIRF implementations run on a single core. If sufficient replications are being carried out to utilize all available cores, comparison of core minute utilization is equivalent to comparison of total computation time. However, a single replication of UBF, ABF or ABF-IR proceeds more quickly due to the parallelization.

ABF-IR and GIRF have computational time scaling quadratically with U in this example, whereas the other methods scale linearly. This is because the number of intermediate steps used, S, grows linearly with U.

The main purpose of this example is not to provide a comparison between the functional capabilities of the methods on interesting scientific problems. It is a toy example without the complexities that the methods are intended to address. This simple example does show clearly the quick decline of PF and the slower declines of GIRF and ABF as dimension increases.

	UBF	ABF	ABF-IR	GIRF	PF	BPF	EnKF
particles, J	—	400	200	1000	100000	20000	10000
bootstrap replications, $\mathcal{I}$	40000	400	200	—	—	—	—
guide simulations, G	—	—	—	50	—	—	—
lookahead lag, L	—	—	—	2	—	—	—
intermediate steps, S	—	—	$U/2$	U	—	—	—
neighborhood, $B_{u,n}$ or block size	{(u-1,n),(u-2,n), (u,n-1),(u,n-2)}			—	—	3	—
forecast mean, $\boldsymbol{\mu}(\boldsymbol{x}, s, t)$	—	—	$\boldsymbol{x}$		—	—	—
measurement mean, $h_{u,n}(x)$	—	—	x		—	—	x
$\tau = \overleftarrow{\mathrm{v}}_{u,n}(V, x)$	—	—	$\sqrt{V}$		—	—	—
$V = \overrightarrow{\mathrm{v}}_{u,n}(\tau, x)$	—	—	τ^2		—	—	τ^2
effort (core mins, $U = 100$)	0.9	2.1	4.2	0.3	0.3	0.2	0.0
effort (core mins, $U = 80$)	1.4	3.2	12.1	0.7	0.6	0.3	0.1
effort (core mins, $U = 60$)	2.2	6.0	32.0	1.9	1.3	0.6	0.2
effort (core mins, $U = 30$)	4.0	11.6	87.1	6.1	3.2	1.3	0.4
effort (core mins, $U = 10$)	7.6	26.1	311.6	47.1	9.1	3.2	1.1

Table 2.1 – Algorithmic settings for the correlated Brownian motions numerical example. Computational effort is measured in core minutes for running one filter, corresponding to a point on Figure 2.1. The time taken for computing a single point using the parallel UBF, ABF and ABF-IR implementations is the effort divided by the number of cores, here 40. The time taken for computing a single point using the single-core GIRF, PF, BPF and EnKF implementations is equal to the effort in core minutes.

2.4.2 A Lorenz-96 example

Our primary motivation for ABF and ABF-IR is application to population dynamics arising in ecological and epidemiological models. Geophysical models provide an alternative situation involving spatiotemporal data analysis. We compare methods on the Lorenz-96 model, a nonlinear chaotic system providing a toy model for global atmospheric circulation (Lorenz, 1996; van Kekem and Sterk, 2018). We consider a stochastic Lorenz-96 model with added Gaussian process noise (Park and Ionides, 2020) defined as the solution to the following system of stochastic differential equations,

$$dX_u(t) = \big\{\big(X_{u+1}(t) - X_{u-2}(t)\big) \cdot X_{u-1}(t) - X_u(t) + F\big\}dt + \sigma_p dB_u(t), \qquad u \in 1\!:\!U. \tag{2.9}$$

We define $X_0 = X_U$, $X_{-1} = X_{U-1}$, and $X_{U+1} = X_1$ so that the U spatial locations are placed on a circle. The terms $\{B_u(t), u \in 1:U\}$ denote U independent standard Brownian motions. F is a forcing constant, and we use the value $F = 8$ which was demonstrated by Lorenz (1996) to induce chaotic behavior. The process noise parameter is set to $\sigma_p = 1$. The system is started with initial state $X_u(0)$ drawn as an independent normal random variable with mean 5 and standard deviation 2 for $u \in 1\!:\!U$. This initialization leads to short transient behavior. Observations are independently made for each dimension at $t_n = n$ for $n \in 1:N$ with Gaussian measurement noise of mean zero and standard deviation $\tau = 1$,

$$Y_{u,n} = X_u(t_n) + \eta_{u,n} \qquad\qquad \eta_{u,n} \sim N(0, \tau^2). \tag{2.10}$$

We used an Euler-Maruyama method for numerical approximation of the sample paths of $\{\boldsymbol{X}(t)\}$, with timestep of 0.005. A simulation from this model is shown in Fig. 2.4.

The ensemble Kalman filter (EnKF) is a widely used filtering method in weather forecasting for high dimensional systems (Evensen and van Leeuwen, 1996). EnKF involves a local Gaussian approximation which is problematic in highly nonlinear systems (Ades and Van Leeuwen, 2015). Methods that make local Gaussian assumptions like EnKF are necessary to scale up to the dimensions of the problems in weather forecasting. Figure 2.3 shows that for a small number of units, the basic particle filter (PF) and GIRF out-perform EnKF. Then, as the number of spatial units increases, the performance of PF rapidly deteriorates whereas GIRF continues to perform well up to a moderate number of units. UBF, ABF, and particularly ABF-IR, scales well despite under-performing EnKF on this example. The additive Gaussian observation and process noise in the Lorenz-96 model is well suited to the approximations involved in EnKF. By contrast, it is less clear how to apply EnKF to discrete population non-Gaussian models such as the measles example in Chapter 4 and how effective the resulting approximations might be.

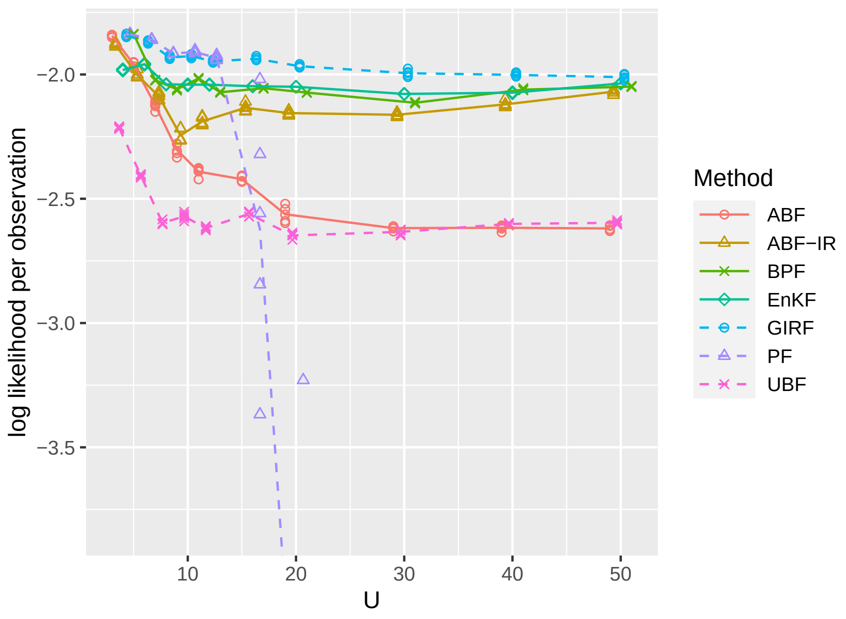


Figure 2.3 – Log likelihood estimates for a Lorenz-96 model of various dimensions. UBF, ABF and ABF-IR are compared with a guided intermediate resampling filter (GIRF), a standard particle filter (PF), a block particle filter (BPF) and an ensemble Kalman filter (EnKF).

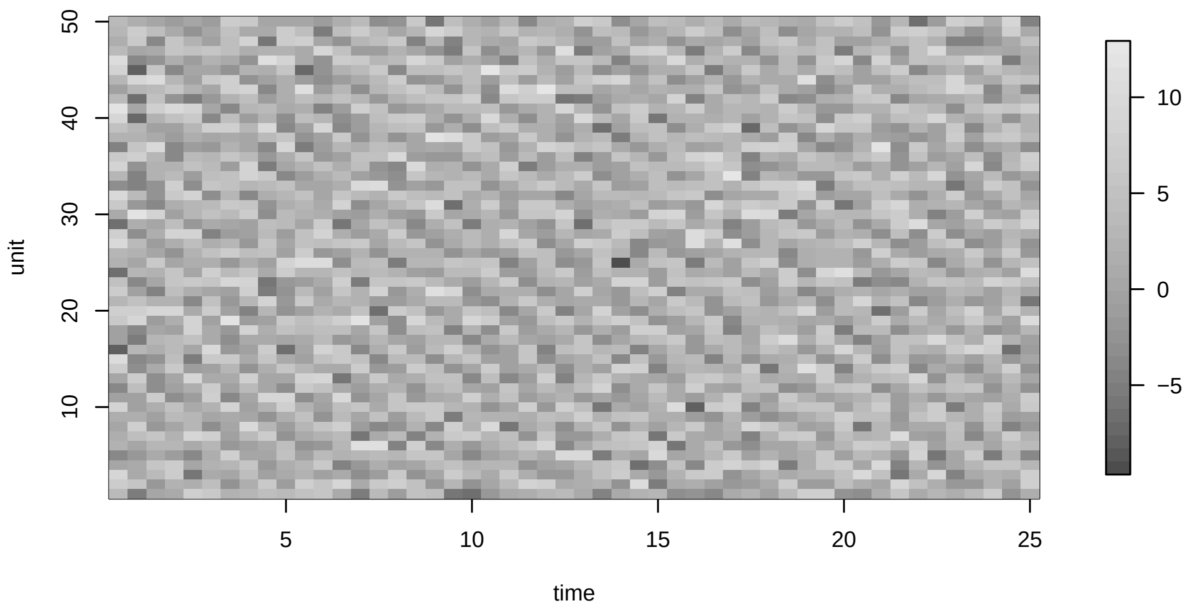


Figure 2.4 – One Lorenz ’96 simulation used for Figure 2.3

	UBF	ABF	ABF-IR	GIRF	PF	EnKF	BPF
particles, J	1	400	200	1000	100000	10000	10000
bootstrap replicates, $\mathcal{I}$	40000	400	200	—	—	—	—
guide simulations, G	—	—	—	50	—	—	—
lookahead lag, L	—	—	—	2	—	—	—
intermediate steps, S	—	—	$U/2$	U	—	—	—
neighborhood, $B_{u,n}$ or block size	{(u,n-1),(u,n-2), (u-1,n),(u-2,n)}			—	—	—	4
forecast mean, $\boldsymbol{\mu}(\boldsymbol{x}, s, t)$	—	—	ODE model		—	—	—
measurement mean, $h_{u,n}(x)$	—	—	x		—	x	—
$\tau = \overleftarrow{\mathrm{v}}_{u,n}(V, x)$	—	—	$\sqrt{V}$		—	—	—
$V = \overrightarrow{\mathrm{v}}_{u,n}(\tau, x)$	—	—	τ^2		—	τ^2	—
effort (core mins, $U = 4$)	34.5	5.2	5.1	2.0	2.0	0.2	0.2
effort (core mins, $U = 6$)	40.9	7.3	8.2	3.0	2.9	0.3	0.3
effort (core mins, $U = 10$)	54.5	11.2	16.3	5.0	4.8	0.5	0.5
effort (core mins, $U = 16$)	75.5	16.8	33.7	8.3	7.6	0.8	0.9
effort (core mins, $U = 50$)	202.3	48.6	174.8	34.4	23.7	2.5	2.7

Table 2.2 – Algorithmic settings for the Lorenz-96 numerical example. Computational effort is measured in core minutes for running one filter, corresponding to a point on Figure Figure 2.3. The time taken for computing a single point using the parallel UBF, ABF and ABF-IR implementations is the effort divided by the number of cores, here 36. The time taken for computing a single point using the single-core GIRF, PF, EnKF and BPF implementations is equal to the effort in core minutes.

2.5 Discussion

The pseudocode presented for the bagged filters describes how the outputs are calculated given the inputs, but does not prescribe details of how these quantities are calculated. There is scope for implementations to trade off memory, computation and communication by varying decisions on how the loops defined in the pseudocode are coded, including decisions on memory over-writing and parallelization. This chapter focuses on the logical structure of the algorithms, leaving room for future research on implementation-specific considerations.

Plug-and-play inference based on sequential Monte Carlo likelihood evaluation has proved successful for investigating highly nonlinear partially observed dynamic systems of low dimension arising in analysis of epidemiological and ecological population dynamics (Bretó, 2018; Pons-Salort and Grassly, 2018; de Cellès et al., 2018; Marino et al., 2019). This chapter develops a methodological extension motivated by the analysis of interacting biological populations. Similar challenges related to nonlinear non-Gaussian dynamic models arise in geophysical modeling. Relative to biological systems, geophysical applications are characterized by a greater number of spatial locations, better mathematical understanding of the underlying processes, and lower stochasticity. From this literature, the locally weighted particle filter of Poterjoy (2016) is perhaps closest to our approach, but the local weights of Poterjoy (2016) are used to construct a localized Kalman gain which is motivated by a Gaussian approximation comparable to EnKF. EnKF arose originally via geophysical research (Evensen and van Leeuwen, 1996) and has since become used more widely applied for inference on SpatPOMP models (Katzfuss et al., 2020; Lei et al., 2010). However, EnKF can fail entirely even on simple POMP models if the structure is sufficiently non-Gaussian. For example, let X_n be a one-dimensional Gaussian random walk, and let Y_n given $X_n = x_n$ be normally distributed with mean 0 and variance x_n^2. The linear filter rule used by EnKF to update the estimate of X_n given Y_n has mean zero for any value of X_n, since X_n and Y_n are uncorrelated. Therefore, the EnKF filter estimate of the latent process remains essentially constant regardless of the data. Models of this form are used in finance to describe stochastic volatility. EnKF could be applied more successfully by modifying model, such as replacing Y_n by $|Y_n|$, but for complex models it may be unclear whether and where such problems are arising. Our results show that there is room for improvement over EnKF on a spatiotemporal epidemiology model, though in our example there is no clear advantage for BF methods over BPF.

Latent state trajectories constructed in our BF algorithms are all generated from the model simulator, appropriately reweighted and resampled, and so are necessarily valid sample paths of the model. For example, spatial smoothness properties of the model through space, or conservation properties where some function of the system remains unchanged through time, are maintained in the BF trajectories. This is not generally true for the block particle filter (since resampling blocks can lead to violations at block boundaries) or for EnKF (since the filter procedure perturbs particles using a linear update rule that cannot respect nonlinear relationships). The practical importance of smoothness and conservation considerations will vary with the system under investigation, but this property of BF gives the scientific investigator one less thing to worry about.

The algorithms UBF, ABF, ABF-IR, GIRF, PF, BPF, and EnKF compared in this article all enjoy the plug-and-play property, facilitating their implementations in general-purpose software. The numerical results for this paper use the `abf`, `abfir`, `girf`, `pfilter`, `bpfilter` and `enkf` methods via the open-source R package `spatPomp` (Asfaw et al., 2021b) that provides a spatiotemporal extension of the R package `pomp` (King et al., 2016). UBF was implemented using `abf` with $J = 1$ particles per replicate. The source code for this paper will be contributed to an open-source scientific archive upon acceptance for publication.

2.6 Appendix for Chapter 2

2.A Proof of Theorem 1

We use the total variation bound in Assumption A1 via the following Proposition 1, which replaces conditioning on $X_{B^c_{u,n}}$ with conditioning on $Y_{B^c_{u,n}}$. The bound in (2.11) could be used in place of Assumption A1.

Proposition 1. *Under the conditions of Assumption A1, (2.2) implies*

$$\left| \int h(x_{u,n}) f_{X_{u,n}|Y_{A_{u,n}}}(x_{u,n} \,|\, y^*_{A_{u,n}})\, dx_{u,n} - \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}}(x_{u,n} \,|\, y^*_{B_{u,n}})\, dx_{u,n} \right| < \epsilon_{\mathrm{A1}} \qquad (2.11)$$

Proof. For notational compactness, we suppress the arguments $x_{u,n}$, $x_{B^c_{u,n}}$, $y^*_{A_{u,n}}$, $y^*_{B_{u,n}}$ matching the subscripts of conditional densities. Using the conditional independence of the measurements given the latent process, we calculate

$$\begin{aligned}
&\left| \int h(x_{u,n}) f_{X_{u,n}|Y_{A_{u,n}}}\, dx_{u,n} - \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}}\, dx_{u,n} \right| \\
&\quad = \left| \int \left\{ \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}, X_{B^c_{u,n}}}\, dx_{u,n} - \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}}\, dx_{u,n} \right\} f_{X_{B^c_{u,n}}\,|\,Y_{A_{u,n}}}\, dx_{B^c_{u,n}} \right| \\
&\quad \leq \int \left| \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}, X_{B^c_{u,n}}}\, dx_{u,n} - \int h(x_{u,n}) f_{X_{u,n}|Y_{B_{u,n}}}\, dx_{u,n} \right| f_{X_{B^c_{u,n}}\,|\,Y_{A_{u,n}}}\, dx_{B^c_{u,n}} \\
&\quad < \int \epsilon_{\mathrm{A1}}\, f_{X_{B^c_{u,n}}\,|\,Y_{A_{u,n}}}\, dx_{B^c_{u,n}} \quad = \quad \epsilon_{\mathrm{A1}}.
\end{aligned}$$

□

Assumption A4 is needed only to ensure that the variance bound in Theorem 1 is essentially $O(UN)$ rather than $O(U^2N^2)$. Both these rates avoid the exponentially increasing variance characterizing the curse of dimensionality. Lower variance than $O(UN)$ cannot be anticipated for any sequential Monte Carlo method since the log likelihood estimate can be written as a sum of UN terms each of which involves its own sequential Monte Carlo calculation.

Proof of Theorem 1. Suppose the quantities $w^M_{u,n,i}$ and $w^P_{u,n,i}$ constructed in Algorithm UBF are considered i.i.d. replicates of jointly defined random variables $w^M_{u,n}$ and $w^P_{u,n}$, for each $(u,n) \in 1\!:\!U \times 1\!:\!N$. Also, write

$$\Delta^{MP}_{u,n} = \frac{1}{\sqrt{\mathcal{I}}}\sum_{i=1}^{\mathcal{I}}\big(w^M_{u,n,i}w^P_{u,n,i} - \mathbb{E}[w^M_{u,n}w^P_{u,n}]\big), \qquad \Delta^P_{u,n} = \frac{1}{\sqrt{\mathcal{I}}}\sum_{i=1}^{\mathcal{I}}\big(w^P_{u,n,i} - \mathbb{E}[w^P_{u,n}]\big),$$

Then, using the delta method (e.g., Section 2.5.3 in Liu (2001)) we find

$$\begin{aligned}
\ell^{\mathrm{MC}}_{u,n} &= \log\left(\frac{\sum_{i=1}^{\mathcal{I}} w^M_{u,n,i}w^P_{u,n,i}}{\sum_{i=1}^{\mathcal{I}} w^P_{u,n,i}}\right)\\
&= \log\left(\mathbb{E}[w^M_{u,n}w^P_{u,n}] + \mathcal{I}^{-1/2}\Delta^{MP}_{u,n}\right) - \log\left(\mathbb{E}[w^P_{u,n}] + \mathcal{I}^{-1/2}\Delta^P_{u,n}\right)\\
&= \log\left(\frac{\mathbb{E}[w^M_{u,n}w^P_{u,n}]}{\mathbb{E}[w^P_{u,n}]}\right) + \mathcal{I}^{-1/2}\left(\frac{\Delta^{MP}_{u,n}}{\mathbb{E}[w^M_{u,n}w^P_{u,n}]} - \frac{\Delta^P_{u,n}}{\mathbb{E}[w^P_{u,n}]}\right) + o_P(\mathcal{I}^{-1/2}) \qquad (2.12)
\end{aligned}$$

The joint distribution of $\big\{(\Delta^{MP}_{u,n}, \Delta^P_{u,n}), (u,n) \in 1\!:\!U \times 1\!:\!N\big\}$ follows a standard central limit theorem as $\mathcal{I} \to \infty$. Each term has mean zero, with covariances uniformly bounded over $(u, n, \tilde{u}, \tilde{n})$ due to Assumption A3. Specifically,

$$\mathrm{Var}\begin{pmatrix}\Delta^{MP}_{u,n}\\ \Delta^P_{u,n}\\ \Delta^{MP}_{\tilde{u},\tilde{n}}\\ \Delta^P_{\tilde{u},\tilde{n}}\end{pmatrix} = \begin{pmatrix}
\mathrm{Var}(w^M_{u,n}w^P_{u,n}) & \mathrm{Cov}(w^M_{u,n}w^P_{u,n}, w^P_{u,n}) & \mathrm{Cov}(w^M_{u,n}w^P_{u,n}, w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}) & \mathrm{Cov}(w^M_{u,n}w^P_{u,n}, w^P_{\tilde{u},\tilde{n}})\\
\mathrm{Cov}(w^M_{u,n}w^P_{u,n}, w^P_{u,n}) & \mathrm{Var}(w^P_{u,n}) & \mathrm{Cov}(w^P_{u,n}, w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}) & \mathrm{Cov}(w^P_{u,n}, w^P_{\tilde{u},\tilde{n}})\\
\mathrm{Cov}(w^M_{u,n}w^P_{u,n}, w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}) & \mathrm{Cov}(w^P_{u,n}, w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}) & \mathrm{Var}(w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}) & \mathrm{Cov}(w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}, w^P_{\tilde{u},\tilde{n}})\\
\mathrm{Cov}(w^M_{u,n}w^P_{u,n}, w^P_{\tilde{u},\tilde{n}}) & \mathrm{Cov}(w^P_{u,n}, w^P_{\tilde{u},\tilde{n}}) & \mathrm{Cov}(w^M_{\tilde{u},\tilde{n}}w^P_{\tilde{u},\tilde{n}}, w^P_{\tilde{u},\tilde{n}}) & \mathrm{Var}(w^P_{\tilde{u},\tilde{n}})
\end{pmatrix}$$

Note that

$$\begin{aligned}\log\left[\frac{\mathbb{E}[w^M_{u,n}w^P_{u,n}]}{\mathbb{E}[w^P_{u,n}]}\right] &= \log\left[\frac{\int f_{Y_{u,n}|X_{u,n}}(y^*_{u,n}\,|\,x_{u,n,i})\,f_{Y_{B_{u,n}}|X_{B_{u,n}}}(y^*_{B_{u,n}}\,|\,x_{B_{u,n}})\,f_{X_{B^+_{u,n}}}(x_{B^+_{u,n}})\,dx_{B^+_{u,n}}}{\int f_{Y_{B_{u,n}}|X_{B_{u,n}}}(y^*_{B_{u,n}}\,|\,x_{B_{u,n}})\,f_{X_{B_{u,n}}}(x_{B_{u,n}})\,dx_{B_{u,n}}}\right]\\ &= \log\left[f_{Y_{u,n}|Y_{B_{u,n}}}(y^*_{u,n}\,|\,y^*_{B_{u,n}})\right],\end{aligned}$$

where $B^+_{u,n} = B_{u,n}\cup(u,n)$. Now, define

$$\Delta^\ell_{u,n} = \left(\frac{\Delta^{MP}_{u,n}}{\mathbb{E}[w^M_{u,n}w^P_{u,n}]} - \frac{\Delta^P_{u,n}}{\mathbb{E}[w^P_{u,n}]}\right)$$

Summing over all $(u,n)\in 1\!:\!U\times 1\!:\!N$, we get

$$\sqrt{\mathcal{I}}\left(\ell^{\mathrm{MC}} - \sum_{(u,n)\in 1:U\times 1:N}\log f_{Y_{u,n}|Y_{B_{u,n}}}(y^*_{u,n}\,|\,y^*_{B_{u,n}})\right) = \sum_{(u,n)\in 1:U\times 1:N}\Delta^\ell_{u,n} + o(1). \qquad (2.13)$$

Now,

$$\mathrm{Cov}(\Delta^\ell_{u,n},\Delta^\ell_{\tilde u,\tilde n}) = \mathrm{Cov}\left(\frac{w^M_{u,n}w^P_{u,n}}{\mathbb{E}[w^M_{u,n}w^P_{u,n}]} - \frac{w^P_{u,n}}{\mathbb{E}[w^P_{u,n}]}, \frac{w^M_{\tilde u,\tilde n}w^P_{\tilde u,\tilde n}}{\mathbb{E}[w^M_{\tilde u,\tilde n}w^P_{\tilde u,\tilde n}]} - \frac{w^P_{\tilde u,\tilde n}}{\mathbb{E}[w^P_{\tilde u,\tilde n}]}\right).$$

Since

$$\left|\frac{w^M_{u,n}w^P_{u,n}}{\mathbb{E}[w^M_{u,n}w^P_{u,n}]} - \frac{w^P_{u,n}}{\mathbb{E}[w^P_{u,n}]}\right| < Q^{2b},$$

we have

$$\left|\mathrm{Cov}(\Delta^\ell_{u,n},\Delta^\ell_{\tilde u,\tilde n})\right| < Q^{4b},$$

implying that

$$\mathrm{Var}\left(\sum_{(u,n)\in 1:U\times 1:N}\Delta^\ell_{u,n}\right) < Q^{4b}U^2N^2. \qquad (2.14)$$

If, in addition, (u,n) and $(\tilde u,\tilde n)\in A_{u,n}$ are sufficiently separated in the sense of Assumption A4, then Lemma 1 shows that Assumption A4 implies

$$\left|\mathrm{Cov}(\Delta^{\ell}_{u,n}, \Delta^{\ell}_{\tilde{u},\tilde{n}})\right| < \epsilon_{\mathrm{A4}} Q^{4b}.$$

The number of insufficiently separated neighbors to (u, n) is bounded by c, and so we obtain

$$\mathrm{Var}\left(\sum_{(u,n)\in S} \Delta^{\ell}_{u,n}\right) < Q^{4b}\, UN(c + \epsilon_{\mathrm{A4}}\,(UN - c)). \tag{2.15}$$

Now we proceed to bound the bias in the Monte Carlo central limit estimator of ℓ. Putting $h(x_{u,n}) = f_{Y_{u,n}|X_{u,n}}(y^{*}_{u,n} \,|\, x_{u,n})$ into Assumption A1, using Assumption A3, gives

$$\left|f_{Y_{u,n}|Y_{B_{u,n}}}(y^{*}_{u,n} \,|\, y^{*}_{B_{u,n}}) - f_{Y_{u,n}|Y_{A_{u,n}}}(y^{*}_{u,n} \,|\, y^{*}_{A_{u,n}})\right| < \epsilon_{\mathrm{A1}} Q.$$

Noting that

$$|a - b| < \delta, \quad a > Q^{-1} \text{ and } b > Q^{-1} \text{ implies } |\log(a) - \log(b)| < \delta Q, \tag{2.16}$$

we find

$$|\log f_{Y_{u,n}|Y_{B_{u,n}}}(y^{*}_{u,n} \,|\, y^{*}_{B_{u,n}}) - \log f_{Y_{u,n}|Y_{A_{u,n}}}(y^{*}_{u,n} \,|\, y^{*}_{A_{u,n}})| < \epsilon_{\mathrm{A1}} Q^{2}. \tag{2.17}$$

Summing over (u, n), we get

$$\left|\ell - \sum_{(u,n)\in S} \log f_{Y_{u,n}|Y_{B_{u,n}}}(y^{*}_{u,n} \,|\, y^{*}_{B_{u,n}})\right| < \epsilon_{\mathrm{A1}} Q^{2} UN. \tag{2.18}$$

Together, the results in (2.13), (2.14), (2.15) and (2.18) confirm the assertions of the theorem. $\square$

2.B Proof of Theorem 2

Lemma 1. *Suppose U and V are random variables with joint density satisfying*

$$\big|f_{V|U}(v\,|\,u) - f_V(v)\big| < \epsilon f_V(v). \tag{2.19}$$

Suppose $|g(U)| < a$ and $|h(V)| < b$ for some real-valued function g and h. Then, $\mathrm{Cov}\big(g(U), h(V)\big) < ab\epsilon$.

Proof. The result is obtained by direct calculation, as follows.

$$\begin{aligned}
\mathrm{Cov}\big(g(U), h(V)\big) &= \mathbb{E}\Big[g(U)\,\mathbb{E}\Big[h(V) - \mathbb{E}[h(V)]\,\Big|\,U\Big]\Big] \\
&= \int\left\{\int g(u)h(v)\big(f_{V|U}(v\,|\,u) - f_V(v)\big)\,dv\right\} f_U(u)\,du \\
&< \int\left\{\int ab\epsilon f_V(v)\,dv\right\} f_U(u)\,du \\
&= ab\epsilon.
\end{aligned}$$

□

Let $g_{\boldsymbol{X}_{0:N},\boldsymbol{X}^P_{1:N}}(\boldsymbol{x}_{0:N}, \boldsymbol{x}^P_{1:N})$ be the joint density of the adapted process and the proposal process,

$$g_{\boldsymbol{X}_{0:N},\boldsymbol{X}^P_{1:N}}(\boldsymbol{x}_{0:N}, \boldsymbol{x}^P_{1:N}) = f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)\prod_{n=1}^{N} f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1},\boldsymbol{Y}_n}(\boldsymbol{x}_n\,|\,\boldsymbol{x}_{n-1}, \boldsymbol{y}^*_n)\ f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}^P_n\,|\,\boldsymbol{x}_{n-1}). \tag{2.20}$$

For $B \subset 1\!:\!U \times 1:N$, define $B^{[m]} = B \cap \big(1\!:\!U \times \{m\}\big)$ and set

$$\gamma_B = \prod_{m=1}^{N} f_{Y_{B^{[m]}}|\boldsymbol{X}_{m-1}}(y^*_{B^{[m]}}\,|\,\boldsymbol{X}_{m-1}), \tag{2.21}$$

using the convention that an empty density $f_{Y_\emptyset}$ evaluates to 1. If we denoting $\mathbb{E}_g$ for expectation for $(\boldsymbol{X}_{0:N}, \boldsymbol{X}^P_{1:N})$ having density $g_{\boldsymbol{X}_{0:N}, \boldsymbol{X}^P_{1:N}}$, (2.21) can be written as

$$\gamma_B = \mathbb{E}_g \Big[f_{Y_B | X_B}(y^*_B \,|\, X^P_B) \,\Big|\, \boldsymbol{X}_{0:N} \Big],$$

so we have

$$\mathbb{E}_g\big[\gamma_B\big] = \mathbb{E}_g \Big[f_{Y_B|X_B}(y^*_B | X^P_B) \Big].$$

Two useful identities are

$$\begin{aligned} f_{X_{u,n}|Y_{A_{u,n}}}(x_{u,n}|y^*_{A_{u,n}}) &= \frac{\mathbb{E}_g\Big[f_{Y_{A_{u,n}}|X_{A_{u,n}}}(y^*_{A_{u,n}} \,|\, X^P_{A_{u,n}})\, f_{X_{u,n}|X^{[n]}_{A_{u,n}}, \boldsymbol{X}_{n-1}}(x_{u,n}|X^P_{A^{[n]}_{u,n}}, \boldsymbol{X}_{n-1})\Big]}{\mathbb{E}_g\Big[f_{Y_{A_{u,n}}|X_{A_{u,n}}}(y^*_{A_{u,n}}|X^P_{A_{u,n}})\Big]}, \\ f_{Y_{u,n}|Y_{A_{u,n}}}(y^*_{u,n}|y^*_{A_{u,n}}) &= \frac{\mathbb{E}_g\big[\gamma_{A^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{A_{u,n}}\big]}. \end{aligned}$$

Proposition 2. *Setting $h(x) = f_{Y_{u,n}|X_{u,n}}(y^*_{u,n}|x)$, assumptions B1 and B3 imply*

$$\left| \frac{\mathbb{E}_g\big[\gamma_{A^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{A_{u,n}}\big]} - \frac{\mathbb{E}_g\big[\gamma_{B^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{B_{u,n}}\big]} \right| < Q\epsilon. \tag{2.22}$$

Proof. Using the non-negativity of all terms to justify interchange of integral and expectation,

$$\begin{aligned} &\int h(x) \mathbb{E}_g\Big[f_{Y_{B_{u,n}}|X_{B_{u,n}}}(y^*_{B_{u,n}}|X^P_{B_{u,n}}) f_{X_{u,n}|X_{B^{[n]}_{u,n}}, \boldsymbol{X}_{n-1}}(x|X^P_{B^{[n]}_{u,n}}, \boldsymbol{X}_{n-1})\Big] dx \\ &= \mathbb{E}_g\Big[\int f_{Y_{u,n}|X_{u,n}}(y^*_{u,n}|x) f_{X_{u,n}|X_{B^{[n]}_{u,n}}, \boldsymbol{X}_{n-1}}(x|X^P_{B^{[n]}_{u,n}}, \boldsymbol{X}_{n-1}) dx \cdot f_{Y_{B_{u,n}}|X_{B_{u,n}}}(y^*_{B_{u,n}}|X^P_{B_{u,n}})\Big] \end{aligned} \tag{2.23}$$

But by the construction of g,

$$f_{X_{u,n}|X_{B^{[n]}_{u,n}},\boldsymbol{X}_{n-1}}(x|X^P_{B^{[n]}_{u,n}},\boldsymbol{X}_{n-1}) = g_{X^P_{u,n}|X^P_{B^{[n]}_{u,n}},\boldsymbol{X}_{n-1}}(x|X^P_{B^{[n]}_{u,n}},\boldsymbol{X}_{n-1})$$

$$= g_{X^P_{u,n}|X^P_{B_{u,n}},\boldsymbol{X}_{n-1}}(x|X^P_{B_{u,n}},\boldsymbol{X}_{n-1}).$$

Thus (2.23) becomes

$$\mathbb{E}_g\Big[\int f_{Y_{u,n}|X_{u,n}}(y^*_{u,n}|x)g_{X^P_{u,n}|X_{B^P_{u,n}},\boldsymbol{X}_{n-1}}(x|X^P_{B_{u,n}},\boldsymbol{X}_{n-1})dx\cdot f_{Y_{B_{u,n}}|X_{B_{u,n}}}(y^*_{B_{u,n}}|X^P_{B_{u,n}})\Big]$$

$$= \mathbb{E}_g\left[\mathbb{E}_g\Big[f_{Y_{u,n}|X_{u,n}}(y^*_{u,n}|X^P_{u,n})|X^P_{B_{u,n}},\boldsymbol{X}_{n-1}\Big]\cdot f_{Y_{B_{u,n}}|X_{B_{u,n}}}(y^*_{B_{u,n}}|X^P_{B_{u,n}})\right]$$

$$= \mathbb{E}_g\Big[f_{Y_{B^+_{u,n}}|X_{B^+_{u,n}}}(y^*_{B^+_{u,n}}|X^P_{B^+_{u,n}})\Big]$$

$$= \mathbb{E}_g\gamma_{B^+_{u,n}}.$$

Applying the same argument for the special case of $B_{u,n} = A_{u,n}$, we substitute into Assumption B1 to complete the proof with the fact that $h < Q$. $\square$

The mixing of the adapted process in Assumption B4 replaces the mixing of the unconditional process in Assumption A4. Though mixing of the adapted process may be hard to check, one may suspect that the adapted process typically mixes more rapidly than the unconditional process. Assumption B4 is needed only to ensure that the variance bound in Theorem 2 is essentially $O(UN)$ rather than $O(U^2N^2)$. Either of these rates avoids the exponentially increasing variance characterizing the curse of dimensionality. Lower variance than $O(UN)$ cannot be anticipated for any sequential Monte Carlo method since the log likelihood estimate can be written as a sum of UN terms each of which involves its own sequential Monte Carlo calculation. The following Proposition 3 gives an implication of Assumption B4.

Proposition 3. *Assumption B4 implies that, if $(\tilde{u},\tilde{n}) \notin C_{u,n}$,*

$$\mathrm{Cov}_g\big(\gamma_{B_{u,n}}, \gamma_{B_{\tilde{u},\tilde{n}}}\big) < \epsilon_{\mathrm{B4}} Q^{|B_{u,n}|+|B_{\tilde{u},\tilde{n}}|}. \tag{2.24}$$

Proof. Write $\gamma = \gamma_{B_{u,n}}$ and $\tilde{\gamma} = \gamma_{B_{\tilde{u},\tilde{n}}}$. Also, write $B = B_{u,n}$, $\tilde{B} = B_{\tilde{u},\tilde{n}}$ and $f_B(x_B^P) = f_{Y_B|X_B}(y_B^* \,|\, x_B^P)$. Then,

$$\begin{aligned}
\mathbb{E}[\gamma\tilde{\gamma}] &= \int \left[\int f_B(x_B^P) g_{X_B^P|\boldsymbol{X}_{0:N}}(x_B^P \,|\, \boldsymbol{x}_{0:N})\, dx_B^P\right] \\
&\qquad \times \left[\int f_{\tilde{B}}(x_{\tilde{B}}^P) g_{X_{\tilde{B}}^P|\boldsymbol{X}_{0:N}}(x_{\tilde{B}}^P \,|\, \boldsymbol{x}_{0:N})\, dx_{\tilde{B}}^P\right] g_{\boldsymbol{X}_{0:N}}(\boldsymbol{x}_{0:N})\, d\boldsymbol{x}_{0:N} \\
&= \int\int f_B(x_B^P)\, f_{\tilde{B}}(x_{\tilde{B}}^P) \Big\{ \int g_{X_B^P|\boldsymbol{X}_{0:N}}(x_B^P \,|\, \boldsymbol{x}_{0:N}) \\
&\qquad \times g_{X_{\tilde{B}}^P|\boldsymbol{X}_{0:N}}(x_{\tilde{B}}^P \,|\, \boldsymbol{x}_{0:N})\, g_{\boldsymbol{X}_{0:N}}(\boldsymbol{x}_{0:N})\, d\boldsymbol{x}_{0:N} \Big\}\, dx_B^P\, dx_{\tilde{B}}^P
\end{aligned} \tag{2.25}$$

Putting the approximate conditional independence requirement of Assumption B4 into (2.25), we have

$$\begin{aligned}
&\Big|\mathbb{E}[\gamma\tilde{\gamma}] - \int f_B(x_B^P)\, f_{\tilde{B}}(x_{\tilde{B}}^P)\, g_{X_B^P X_{\tilde{B}}^P|\boldsymbol{X}_{0:N}}(x_B^P, x_{\tilde{B}}^P \,|\, \boldsymbol{x}_{0:N})\, g_{\boldsymbol{X}_{0:N}}(\boldsymbol{x}_{0:N})\, dx_B^P\, dx_{\tilde{B}}^P\, d\boldsymbol{x}_{0:N}\Big| \\
&\qquad < (1/2)\, \epsilon_{\mathrm{B4}}\, Q^{|B|+|\tilde{B}|}.
\end{aligned}$$

This gives

$$\Big|\mathbb{E}[\gamma\tilde{\gamma}] - \int f_B(x_B^P)\, f_{\tilde{B}}(x_{\tilde{B}}^P)\, g_{X_B^P X_{\tilde{B}}^P}(x_B^P, x_{\tilde{B}}^P)\, dx_B^P\, dx_{\tilde{B}}^P\Big| < (1/2)\, \epsilon_{\mathrm{B4}}\, Q^{|B|+|\tilde{B}|}. \tag{2.26}$$

Then, using the approximate unconditional independence requirement of Assumption B4 combined with the triangle inequality, (2.26) implies

$$\left|\mathbb{E}[\gamma\tilde{\gamma}] - \int f_B(x_B^P)\, f_{\tilde{B}}(x_{\tilde{B}}^P)\, g_{X_B^P}(x_B^P)\, g_{X_{\tilde{B}}^P}(x_{\tilde{B}}^P)\, dx_B^P\, dx_{\tilde{B}}^P\right| < \epsilon_{\mathrm{B4}}\, Q^{|B|+|\tilde{B}|}. \tag{2.27}$$

We can rewrite (2.27) as

$$\left|\, \mathbb{E}[\gamma\tilde{\gamma}] - \mathbb{E}[\gamma]\, \mathbb{E}[\tilde{\gamma}] \,\right| < \epsilon_{\mathrm{B4}}\, Q^{|B|+|\tilde{B}|}, \tag{2.28}$$

proving the proposition. □

Assumption B5 is needed to ensure the stability of the Monte Carlo approximation to the adapted process. It ensures that any error due to finite Monte Carlo sample size has limited consequences at sufficiently remote time points. One could instead propose a bound that decreases exponentially with K, but that is not needed for the current purposes. The following Proposition 4 is useful for taking advantage of Assumption B5.

Proposition 4. *Suppose that f is a non-negative function and that for some $\epsilon > 0$,*

$$|f(x) - f(x')| < \epsilon f(x')$$

holds for all x, x'. Then for any two probability distributions where the expectations are denoted by $\mathbb{E}_1$ and $\mathbb{E}_2$ and for any random variable X, we have

$$|\mathbb{E}_1 f(X) - \mathbb{E}_2 f(X)| \le \epsilon\, \mathbb{E}_2 f(X).$$

Proof. Let the two probability laws be denoted by P_1 and P_2. We have

$$\begin{aligned}
|\mathbb{E}_1 f(X) - \mathbb{E}_2 f(X)| &= \left| \int f(x) P_1(dx) - \int f(x') P_2(dx') \right| \\
&\le \left| \int\int f(x) P_1(dx) P_2(dx') - \int\int f(x') P_1(dx) P_2(dx') \right|. \\
&\le \int\int |f(x) - f(x')| P_1(dx) P_2(dx') \\
&\le \int \epsilon f(x') P_2(dx') = \epsilon\, \mathbb{E}_2 f(X).
\end{aligned}$$

□

Assumption B6 controls the Monte Carlo error for a single time interval on a single bootstrap replicate. In the case $S = 1$, ABF-IR becomes ABF and this assumption is one of many alternatives for bounding error from importance sampling. The purpose behind the selection of Assumption B6 is to draw on the results of Park and Ionides (2020) for intermediate resampling, and our assumption is a restatement of their Theorem 2. When $S = 1$, the curse of dimensionality for importance sampling has the consequence that C_0 grows exponentially with U. However, Park and Ionides (2020) showed that setting $S = U$ can lead to situations where $C_0(U, N, S)$ in Assumption B6 grows polynomially with U. Here, we do not place requirements concerning the dependence of C_0 on U, N and S since our immediate concern is a limit where $\mathcal{I}$ and J increase. Nevertheless, the numerical results are consistent with the theoretical and empirical results obtained for intermediate resampling in the context of particle filtering by Park and Ionides (2020).

The Monte Carlo conditional independence required by Assumption B7 would hold for ABF-IR if the guide variance $V_{u,n,i}$ were calculated using an independent set of guide simulations to those used for evaluating the measurement weights $w^M_{u,n,i,j}$. For numerical efficiency, the ABF-IR algorithm implemented here constructs a shared pool of simulations for both purposes rather than splitting the pool up between them, in the expectation that the resulting minor violation of Assumption B7 has negligible impact.

Proof of Theorem 2. First, we set up some notation. For $B_{u,n}$ and $w^M_{u,n,i,j}$ constructed by ABF-IR, define

$$\gamma^{MC,i}_{B_{u,n}} = \prod_{m=1}^{n}\left[\frac{1}{J}\sum_{j=1}^{J}\prod_{(\tilde{u},m)\in B^{[m]}_{u,n}} w^M_{\tilde{u},m,i,j}\right] \quad \text{and} \quad \bar{\gamma}^{MC}_{B_{u,n}} = \frac{1}{\mathcal{I}}\sum_{i=1}^{\mathcal{I}}\gamma^{MC,i}_{B_{u,n}}. \tag{2.29}$$

The Monte Carlo conditional likelihoods output by ABF-IR can be written as

$$\ell^{\text{MC}}_{u,n} = \log\bar{\gamma}^{\text{MC}}_{B^+_{u,n}} - \log\bar{\gamma}^{\text{MC}}_{B_{u,n}}. \tag{2.30}$$

We proceed with a similar argument to the proof of Theorem 1. Since $\gamma^{MC,i}_{B_{u,n}}$ are i.i.d. for $i\in 1\!:\!\mathcal{I}$, we can suppose they are replicates of a Monte Carlo random variable $\gamma^{MC}_{B_{u,n}}$. We define

$$\Delta^+_{u,n} = \frac{1}{\sqrt{\mathcal{I}}}\sum_{i=1}^{\mathcal{I}}\big(\gamma^{MC,i}_{B^+_{u,n}} - \mathbb{E}[\gamma^{MC}_{B^+_{u,n}}]\big), \qquad \Delta_{u,n} = \frac{1}{\sqrt{\mathcal{I}}}\sum_{i=1}^{\mathcal{I}}\big(\gamma^{MC,i}_{B_{u,n}} - \mathbb{E}[\gamma^{MC}_{B_{u,n}}]\big).$$

The same calculation as (2.12) gives

$$\ell^{\text{MC}}_{u,n} \;=\; \log\left(\frac{\mathbb{E}[\gamma^{MC}_{B^+_{u,n}}]}{\mathbb{E}[\gamma^{MC}_{B_{u,n}}]}\right) + \mathcal{I}^{-1/2}\left(\frac{\Delta^+_{u,n}}{\mathbb{E}[\gamma^{MC}_{B^+_{u,n}}]} - \frac{\Delta_{u,n}}{\mathbb{E}[\gamma^{MC}_{B_{u,n}}]}\right) + o_P(\mathcal{I}^{-1/2}) \tag{2.31}$$

The joint distribution of $\big\{(\Delta^+_{u,n},\Delta_{u,n}),(u,n)\in 1\!:\!U\times 1\!:\!N\big\}$ follows a standard central limit theorem as $\mathcal{I}\to\infty$. Each term has mean zero, with variances and covariances uniformly bounded over $(u,n,\tilde{u},\tilde{n})$ due to Assumption B3. From Proposition 2, using the same reasoning as (2.17),

$$\left|\log\left(\frac{\mathbb{E}_g\big[\gamma_{A^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{A_{u,n}}\big]}\right) - \log\left(\frac{\mathbb{E}_g\big[\gamma_{B^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{B_{u,n}}\big]}\right)\right| < \epsilon_{\text{B1}}Q^2. \tag{2.32}$$

Now we use Lemma 2 and (2.16) to obtain

$$\begin{aligned}
&\left|\log\left(\frac{\mathbb{E}_g\big[\gamma_{B^+_{u,n}}\big]}{\mathbb{E}_g\big[\gamma_{B_{u,n}}\big]}\right) - \log\left(\frac{\mathbb{E}\big[\gamma^{MC}_{B^+_{u,n}}\big]}{\mathbb{E}\big[\gamma_{B_{u,n}}\big]}\right)\right| \\
&\le \Big|\log\mathbb{E}_g\big[\gamma_{B^+_{u,n}}\big] - \log\mathbb{E}\big[\gamma_{B^+_{u,n}}\big]\Big| + \Big|\log\mathbb{E}_g\big[\gamma_{B_{u,n}}\big] - \log\mathbb{E}\big[\gamma_{B_{u,n}}\big]\Big| \\
&< 2Q^{2b}\big(\epsilon_{\mathrm{B5}} + (K + d_{\max})\epsilon_{\mathrm{B6}}\big). \qquad (2.33)
\end{aligned}$$

The proof of the central limit result in (2.7) is completed by combining (2.31), (2.32) and (2.33). To show (2.8) we check that $\Delta_{u,n}$ and $\Delta_{\tilde{u},\tilde{n}}$ are weakly correlated when (u,n) and $(\tilde{u},\tilde{n})$ are sufficiently separated. By the same reasoning as the proof of Theorem 1, it is sufficient to show that $\gamma^{MC}_{B_{u,n}}$ and $\gamma^{MC}_{B_{\tilde{u},\tilde{n}}}$ are weakly correlated. These Monte Carlo quantities approximate $\gamma_{B_{u,n}}(\boldsymbol{X}_{0:n-1})$ and $\gamma_{B_{\tilde{u},\tilde{n}}}(\boldsymbol{X}_{0:\tilde{n}-1})$ with $\boldsymbol{X}$ drawn from g. Let us suppose $n \ge \tilde{n}$, and write $d_{u,n} = n - \inf_{(v,m)\in B_{u,n}} m$. First, we consider the situation $n - \tilde{n} > K + d_{u,n}$, in which case we can use the Markov property to give

$$\mathrm{Cov}\big(\gamma^{MC}_{B_{u,n}}, \gamma^{MC}_{B_{\tilde{u},\tilde{n}}}\big) < \mathbb{E}\big[\gamma^{MC}_{B_{\tilde{u},\tilde{n}}}\big] \sup_{\boldsymbol{x}} \Big\{\mathbb{E}\big[\gamma^{MC}_{B_{u,n}} \big| \boldsymbol{X}^A_{n-d_{u,n}-K,1} = \boldsymbol{x}\big] - \mathbb{E}\big[\gamma^{MC}_{B_{u,n}}\big]\Big\} \qquad (2.34)$$

Then, the triangle inequality followed by applications of Assumption B5 and Lemma 2 gives

$$\begin{aligned}
&\Big|\mathbb{E}\big[\gamma^{MC}_{B_{u,n}} \big| \boldsymbol{X}^A_{n-d_{u,n}-K,1} = \boldsymbol{x}\big] - \mathbb{E}\big[\gamma^{MC}_{B_{u,n}}\big]\Big| \\
&\quad \le \Big|\mathbb{E}_g\big[\gamma_{B_{u,n}} \big| \boldsymbol{X}_{n-d_{u,n}-K} = \boldsymbol{x}\big] - \mathbb{E}_g\big[\gamma_{B_{u,n}}\big]\Big| \\
&\qquad + \Big|\mathbb{E}\big[\gamma^{MC}_{B_{u,n}} \big| \boldsymbol{X}^A_{n-d_{u,n}-K,1} = \boldsymbol{x}\big] - \mathbb{E}_g\big[\gamma_{B_{u,n}} \big| \boldsymbol{X}_{n-d_{u,n}-K} = \boldsymbol{x}\big]\Big| \\
&\qquad + \Big|\mathbb{E}\big[\gamma^{MC}_{B_{u,n}}\big] - \mathbb{E}_g\big[\gamma_{B_{u,n}}\big]\Big| \\
&\quad \le Q^b\Big(2\epsilon_{\mathrm{B5}} + 2(K + d_{u,n})\epsilon_{\mathrm{B6}}\Big) \qquad (2.35)
\end{aligned}$$

Putting (2.35) into (2.34), we get

$$\mathrm{Cov}(\gamma^{MC}_{B_{u,n}}, \gamma^{MC}_{B_{\tilde{u},\tilde{n}}}) < Q^{2b}\Big(2\epsilon_{\mathrm{B5}} + 2(K + d_{u,n})\epsilon_{\mathrm{B6}}\Big). \tag{2.36}$$

Now we address the situation $n - \tilde{n} \leq K + d_{u,n}$. We apply Lemma 2 on the union $B_{u,n} \cup B_{\tilde{u},\tilde{n}}$ for which the temporal depth is bounded by $d \leq K + d_{u,n} + d_{\tilde{u},\tilde{n}}$. This gives

$$\Big|\mathbb{E}[\gamma^{MC}_{B_{u,n}}\gamma^{MC}_{B_{\tilde{u},\tilde{n}}}] - \mathbb{E}_g[\gamma_{B_{u,n}}\gamma_{B_{\tilde{u},\tilde{n}}}]\Big| < Q^{2b}\Big((2K + d_{u,n} + d_{\tilde{u},\tilde{n}})\epsilon_{\mathrm{B6}} + \epsilon_{\mathrm{B5}}\Big). \tag{2.37}$$

From Proposition 3, if $(\tilde{u},\tilde{n}) \notin C_{u,n}$,

$$\mathrm{Cov}_g(\gamma_{B_{u,n}}, \gamma_{B_{\tilde{u},\tilde{n}}}) < \epsilon_{\mathrm{B4}} Q^{2b}. \tag{2.38}$$

Now, we establish that $\mathrm{Cov}(\gamma^{MC}_{B_{u,n}}, \gamma^{MC}_{B_{\tilde{u},\tilde{n}}})$ is close to $\mathrm{Cov}_g(\gamma_{B_{u,n}}, \gamma_{B_{\tilde{u},\tilde{n}}})$.

$$\begin{aligned}
&\Big|\mathrm{Cov}(\gamma^{MC}_{B_{u,n}}, \gamma^{MC}_{B_{\tilde{u},\tilde{n}}}) - \mathrm{Cov}_g(\gamma_{B_{u,n}}, \gamma_{B_{\tilde{u},\tilde{n}}})\Big| \\
&\qquad \leq \Big|\mathbb{E}[\gamma^{MC}_{B_{u,n}}\gamma^{MC}_{B_{\tilde{u},\tilde{n}}}] - \mathbb{E}_g[\gamma_{B_{u,n}}\gamma_{B_{\tilde{u},\tilde{n}}}]\Big| \\
&\qquad\qquad + \Big|\mathbb{E}[\gamma^{MC}_{B_{u,n}}](\mathbb{E}[\gamma^{MC}_{B_{\tilde{u},\tilde{n}}}] - \mathbb{E}_g[\gamma_{B_{\tilde{u},\tilde{n}}}])\Big| \\
&\qquad\qquad + \Big|\mathbb{E}[\gamma_{B_{\tilde{u},\tilde{n}}}](\mathbb{E}[\gamma^{MC}_{B_{u,n}}] - \mathbb{E}_g[\gamma_{B_{u,n}}])\Big| \\
&\qquad < Q^{2b}\Big((2K + d_{u,n} + d_{\tilde{u},\tilde{n}})\epsilon_{\mathrm{B6}} + \epsilon_{\mathrm{B5}} + 2(\epsilon_{\mathrm{B5}} + (K + d_{\max})\epsilon_{\mathrm{B6}})\Big). \\
&\qquad < Q^{2b}\Big(3\epsilon_{\mathrm{B5}} + 4(K + d_{\max})\,\epsilon_{\mathrm{B6}}\Big)
\end{aligned} \tag{2.39}$$

Using (2.39) together with (2.38) to bound the $UN(UN - c)$ off-diagonal covariance terms completes the derivation of (2.8).

$\square$

Lemma 2. *Suppose Assumptions B3, B5, B6 and B7. Suppose the number of particles J exceeds the requirement for B6. If we write $d_B = \max_{(u_1,n_1),(u_2,n_2)\in B} |n_1 - n_2|$ for $B \subset 1\!:\!U \times 1\!:\!N$, then for any B,*

$$\Big| \mathbb{E}\big[\gamma_B^{MC} \big| \boldsymbol{X}^A_{n-d_B-K,1} = \boldsymbol{x}\big] - \mathbb{E}_g\big[\gamma_B \big| \boldsymbol{X}_{n-d_B-K} = \boldsymbol{x}\big] \Big| < Q^{|B|}(K+d_B)\epsilon_{\mathrm{B6}}, \quad \forall \boldsymbol{x} \in \mathbb{X}^U,$$

and

$$\Big| \mathbb{E}\big[\gamma_B^{MC}\big] - \mathbb{E}_g\big[\gamma_B\big] \Big| < Q^{|B|}(\epsilon_{\mathrm{B5}} + (K+d_B)\epsilon_{\mathrm{B6}}). \tag{2.40}$$

Proof. Suppose that $\max_{(u',n')\in B} n' = n$. Define $\eta_n(\boldsymbol{x}_n) = 1$ and, for $0 \le m \le n-1$,

$$\eta_m(\boldsymbol{x}_m) = \mathbb{E}_g\left[\prod_{k=m+1}^{n} \gamma_{B[k]} \,\Big|\boldsymbol{X}_m = \boldsymbol{x}_m \right]. \tag{2.41}$$

We have a recursive identity

$$\eta_m(\boldsymbol{X}_m) = \mathbb{E}_g\Big[\gamma_{B[m+1]}\, \eta_{m+1}(\boldsymbol{X}_{m+1}) \Big| \boldsymbol{X}_m\Big]. \tag{2.42}$$

By taking the expectation of (2.41), we have

$$\mathbb{E}_g\big[\eta_0(\boldsymbol{X}_0)\big] = \mathbb{E}_g\big[\gamma_B\big]. \tag{2.43}$$

Note that g has marginal density $f_{\boldsymbol{X}_0}$ for $\boldsymbol{X}_0$. We analyze an ABF-IR approximation to (2.42). The function $\eta_{m+1}(\boldsymbol{x})$ is not in practice computationally available for evaluation via ABF-IR, but the recursion nevertheless leads to a useful bound. Let $\boldsymbol{X}^{\mathrm{A}}_{m+1}[j](\boldsymbol{x}_m)$ correspond to the variable $\boldsymbol{X}^{IR}_{m+1,S,1,j}$ constructed by ABF-IR conditional on $\boldsymbol{X}^{\mathrm{A}}_{m,1} = \boldsymbol{x}_m$. Equivalently, $\boldsymbol{X}^{\mathrm{A}}_{m+1}[j](\boldsymbol{x}_m)$ matches the variable $\boldsymbol{X}^{\mathrm{A}}_{m+1,1}$ in ABF-IR if the assignment $\boldsymbol{X}^{\mathrm{A}}_{m+1,1} = \boldsymbol{X}^{IR}_{m+1,S,1,1}$ is replaced by $\boldsymbol{X}^{\mathrm{A}}_{m+1,1} = \boldsymbol{X}^{IR}_{m+1,S,1,j}$ conditional on $\boldsymbol{X}^{\mathrm{A}}_{m,1} = \boldsymbol{x}_m$. We define an approximation error $e_m(\boldsymbol{x}_m)$ by

$$\eta_m(\boldsymbol{x}_m) = \frac{1}{J}\sum_{j=1}^{J} f_{Y_{B[m+1]}|\boldsymbol{X}_m}(y^*_{B[m+1]} \mid \boldsymbol{x}_m)\, \eta_{m+1}\big(\boldsymbol{X}^{\mathrm{A}}_{m+1}[j](\boldsymbol{x}_m)\big) + e_m(\boldsymbol{x}_m). \tag{2.44}$$

From Assumptions B3 and B6, $\mathbb{E}\big|e_m(\boldsymbol{x}_m)\big| < \epsilon_{\mathrm{B6}}\, Q^{\left|B^{[m+1:n]}\right|}$ uniformly over $\boldsymbol{x}_m$, Thus, setting $r_m = \mathbb{E}|e_m(\boldsymbol{X}^{A}_{m,1})|$, we have

$$r_m < \epsilon_{\mathrm{B6}}\, Q^{\left|B^{[m+1:n]}\right|}. \tag{2.45}$$

Now, setting $K' = K + d_{u,n}$, we commence to prove inductively that, for $n - K' \le m \le n$,

$$\left|\eta_{n-K'}(\boldsymbol{x}) - \mathbb{E}\Big[\eta_m(\boldsymbol{X}^{A}_{m,1}) \prod_{k=n-K'+1}^{m} f_{Y_{B[k]}|\boldsymbol{X}_{k-1}}(y^*_{B[k]} \mid \boldsymbol{X}^{A}_{k-1,1}) \Big| \boldsymbol{X}^{A}_{n-K',1} = \boldsymbol{x}\Big]\right| < (m-n+K')\epsilon_{\mathrm{B6}}\, Q^{|B|}. \tag{2.46}$$

First, suppose that (2.46) holds for m. From (2.44) and (2.45),

$$\left|\eta_m(\boldsymbol{x}_m) - \mathbb{E}\left[\frac{1}{J}\sum_{j=1}^{J} f_{Y_{B[m+1]}|\boldsymbol{X}_m}(y^*_{B[m+1]} \mid \boldsymbol{x}_m)\, \eta_{m+1}\big(\boldsymbol{X}^{\mathrm{A}}_{m+1}[j](\boldsymbol{x}_m)\big)\right]\right| < \epsilon_{\mathrm{B6}}\, Q^{\left|B^{[m+1:n]}\right|}. \tag{2.47}$$

Since the particles are exchangeable, the expectation of the mean of J particles can be replaced with the expectation of the first particle. Plugging in $\boldsymbol{x}_m = \boldsymbol{X}^{A}_{m,1}$ gives us

$$\left|\eta_m(\boldsymbol{X}^{A}_{m,1}) - f_{Y_{B[m+1]}|\boldsymbol{X}_m}(y^*_{B[m+1]} \mid \boldsymbol{X}^{A}_{m,1})\mathbb{E}\left[\eta_{m+1}(\boldsymbol{X}^{A}_{m+1,1})\Big|\boldsymbol{X}^{A}_{m,1}\right]\right| < \epsilon_{\mathrm{B6}}\, Q^{\left|B^{[m+1:n]}\right|} \tag{2.48}$$

Putting (2.48) into (2.46), for $m \le n$, and taking an iterated expectation with respect to $\boldsymbol{X}^A_{m,1}$, we find that (2.46) holds also for $m+1$. Since (2.46) holds trivially for $m = n - K'$, it holds for $n - K' \le m \le n$ by induction. Then, noting $\eta_n(\boldsymbol{x}) = 1$, we have from (2.46) that

$$\left| \eta_{n-K'}(x) - \mathbb{E}\left[\prod_{k=n-K'+1}^{n} f_{Y_{B[k]} \mid \boldsymbol{X}_{k-1}}(y^*_{B[k]} \mid \boldsymbol{X}^A_{k-1,1}) \middle| \boldsymbol{X}^A_{n-K',1} = x \right] \right| < K' \epsilon_{\mathrm{B6}}\, Q^{|B|}.$$

Integrating the above inequality over $\boldsymbol{x}$ with respect to the law of $\boldsymbol{X}^A_{n-K',1}$, we obtain

$$\left| \mathbb{E}[\eta_{n-K'}(\boldsymbol{X}^A_{n-K',1})] - \mathbb{E}\left[\prod_{k=n-K'+1}^{n} f_{Y_{B[k]} \mid \boldsymbol{X}_{k-1}}(y^*_{B[k]} \mid \boldsymbol{X}^A_{k-1,1}) \right] \right| < K' \epsilon_{\mathrm{B6}}\, Q^{|B|}. \tag{2.49}$$

But under Assumption B7, we have

$$\mathbb{E}\left[\prod_{k=n-K'+1}^{n} f_{Y_{B[k]} \mid \boldsymbol{X}_{k-1}}(y^*_{B[k]} \mid \boldsymbol{X}^A_{k-1,1}) \right] = \mathbb{E}[\gamma^{MC}_B]. \tag{2.50}$$

Assumption B5 says

$$|\eta_{n-K'}(\boldsymbol{x}^{(1)}_{n-K'}) - \eta_{n-K'}(\boldsymbol{x}^{(2)}_{n-K'})| < \epsilon_{\mathrm{B5}}\, \eta_{n-K'}(\boldsymbol{x}^{(2)}_{n-K'}). \tag{2.51}$$

Application of Proposition 4 to (2.51) gives

$$\left| \mathbb{E}_g\big[\eta_{n-K'}(\boldsymbol{X}_{n-K'})\big] - \mathbb{E}\big[\eta_{n-K'}(\boldsymbol{X}^A_{n-K',1})\big] \right| < \epsilon_{\mathrm{B5}}\, \mathbb{E}_g\big[\eta_{n-K'}(\boldsymbol{X}_{n-K'})\big] < \epsilon_{\mathrm{B5}}\, Q^{|B|}. \tag{2.52}$$

Combining (2.49), (2.50), and (2.52) completes the proof of Lemma 2. $\square$

2.C A generalization to models without latent unit structure

Variations of the algorithms in the main text apply when there is no latent unit structure. In this case, the observation vector $\boldsymbol{Y}_n = (Y_{1,n}, \ldots, Y_{U,n})$ consists of a collection of measurements on a general latent vector X_n. We may have the structure that $Y_{1,n}, \ldots, Y_{U,n}$ are conditionally independent given X_n, but even this is not essential to the approach. This is most readily seen in the context of the unadapted bagged filter, giving rise to the generalized unadapted bagged filter (G-UBF) algorithm defined as follows.

```
**Algorithm G-UBF (Generalized unadapted bagged filter).**
input:
Simulator for $f_{X_n|X_{n-1}}(x_n|x_{n-1})$
Evaluator for $f_{Y_{u,n}|X_n}(y^*_{u,n} \,|\, x_n)$
Number of bootstrap filters, $\mathcal{I}$
Neighborhood structure, $B_{u,n}$, for $u \in 1\,{:}\,U$ and $n \in 1\,{:}\,N$
Data, $y^*_{u,n}$ for $u \in 1\,{:}\,U$ and $n \in 1\,{:}\,N$
output:
Log likelihood estimate, $\ell^{\mathrm{MC}} = \sum_{n=1}^{N} \sum_{u=1}^{U} \ell^{\mathrm{MC}}_{u,n}$

For $i$ in $1\,{:}\,\mathcal{I}$
  simulate $X^{\mathrm{sim}}_{n,i}$ from the dynamic model, for $n \in 1\,{:}\,N$
End For
Prediction weights, $w^P_{u,n,i} = f_{Y_{B_{u,n}}|X_{1:n}}(y^*_{B_{u,n}} \,|\, X^{\mathrm{sim}}_{1:n,i})$
Measurement weights, $w^M_{u,n,i} = f_{Y_{u,n}|X_n,Y_{B_{u,n}}}(y^*_{u,n} \,|\, X^{\mathrm{sim}}_{n,i}, y^*_{B_{u,n}})$
Conditional log likelihood estimate, $\ell^{\mathrm{MC}}_{u,n} = \log\Big(\sum_{i=1}^{\mathcal{I}} w^M_{u,n,i}\, w^P_{u,n,i}\Big) - \log\Big(\sum_{\tilde{i}=1}^{\mathcal{I}} w^P_{u,n,\tilde{i}}\Big)$
```

The algorithm G-UBF operates on an arbitrary POMP model. G-UBF therefore provides a potential approach to extending methodologies from SpatPOMP models to models that have some similarity to a SpatPOMP without formally meeting the definition. For example, there may be collections of interacting processes at different spatial scales in a spatiotemporal system. Alternatively, the potential outcomes of the latent process may vary between spatial units, such as when modeling interactions between terrestrial and aquatic ecosystems. We do not further explore G-UBF here.

2.D Adapted simulation for an Euler approximation

We investigate the adapted simulation process by considering a continuous-time limit where it becomes a diffusion process. We find that adapted simulation can effectively track the latent process when the measurement error is on an appropriate scale. However, when the measurement error is large compared to the latent process noise, adapted simulation can fail in situations where filtering succeeds. We work with a one-dimensional POMP model having a latent process constructed as an Euler approximation,

$$X_{n+1} \quad = \quad X_n + \mu(X_n)\delta + \sigma\sqrt{\delta}\epsilon_{n+1}, \tag{2.53}$$

which provides a numerical solution to a one-dimensional stochastic differential equation,

$$dX(t) \quad = \quad \mu\big(X(t)\big)\,dt + \sigma\,dU(t),$$

where $\{U(t)\}$ is a standard Brownian motion. We will consider several different measurement processes.

2.E Measurement error on the same scale as the process noise

Here, we consider the measurement model

$$Y_{n+1} = \mu(X_n)\delta + \sigma\sqrt{\delta}\epsilon_{n+1} + \tau\sqrt{\delta}\,\eta_{n+1}. \tag{2.54}$$

This is an approximation to the increment $Y(t+\delta)-Y(t)$ of a continuous time measurement model

$$dY(t) = dX(t) + \tau\, dV(t), \tag{2.55}$$

where $\{V(t)\}$ is a standard Brownian motion independent of $\{U(t)\}$. The measurement model (2.55) makes inference on $X(t)$ given $Y(t)$ a continuous time version of the filtering problem. A feature of this model is that $Y(t)$ does not directly track the level of the state, since the solution with initial conditions $Y(t_0) = X(t_0)$ and $V(t_0) = 0$ is

$$Y(t) = X(t) + \tau V(t).$$

The measurement error, $\tau V(t)$, has variance $\tau^2 t$ that increases with t. However, under appropriate conditions, information on changes in $\{X(t)\}$ obtained via $\{Y(t)\}$ are enough to track $X(t)$ indirectly via the filtering equations. For the POMP given by (2.53) and (2.54), we can calculate exactly the adapted simulation distribution $f_{X_{n+1}|Y_{n+1},X_n}$. It is convenient to work conditionally on X_n, allowing us to treat X_n and $\mu(X_n)$ as constants, with X_{n+1} and Y_{n+1} therefore being jointly normally distributed. A Gaussian distribution calculation then gives the conditional moments. First, we find

$$\begin{aligned}
\mathbb{E}[X_{n+1}|Y_{n+1},X_n] &= X_n + \mu(X_n)\delta + \mathbb{E}[\sigma\sqrt{\delta}\epsilon_{n+1} \,|\, \sigma\sqrt{\delta}\epsilon_{n+1} + \tau\sqrt{\delta}\eta_{n+1}] \\
&= X_n + \mu(X_n)\delta + \frac{\sigma^2}{\sigma^2+\tau^2}(\sigma\sqrt{\delta}\epsilon_{n+1} + \tau\sqrt{\delta}\eta_{n+1}) \\
&= X_n + \mu(X_n)\delta + \frac{\sigma^2}{\sigma^2+\tau^2}(Y_{n+1} - \mu(X_n)\delta).
\end{aligned}$$

Then,

$$\begin{aligned}
\mathrm{Var}[X_{n+1}\,|\,Y_{n+1},X_n] &= \mathrm{Var}[\sigma\sqrt{\delta}\epsilon_{n+1}\,|\,\sigma\sqrt{\delta}\epsilon_{n+1}+\tau\sqrt{\delta}\eta_{n+1}] \\
&= \sigma^2\delta-\frac{\sigma^4\delta^2}{\sigma^2\delta+\tau^2\delta} \\
&= \delta\frac{\sigma^2\tau^2}{\sigma^2+\tau^2}.
\end{aligned}$$

Call the adapted simulation process $\{A_n, n=1,2,\dots\}$, defined conditionally on $\{Y_n, n=1,2,\dots\}$. We see from the above calculation that A_n can be constructed by the recursion

$$\begin{aligned}
A_{n+1} &= A_n+\mu(A_n)\delta+\frac{\sigma^2}{\sigma^2+\tau^2}\Big(\mu(X_n)\delta+\sigma\sqrt{\delta}\,\epsilon_{n+1}+\tau\sqrt{\delta}\,\eta_{n+1}-\mu(A_n)\,\delta\Big) \\
&\qquad +\frac{\sigma\tau}{\sqrt{\sigma^2+\tau^2}}\sqrt{\delta}\,\zeta_{n+1}
\end{aligned}$$

where $\{\zeta_n\}$ is an iid standard normal sequence independent of $\{\epsilon_n,\eta_n\}$. To study how well the adapted simulation tracks $\{X_n\}$, we subtract X_{n+1} from both sides to get

$$\begin{aligned}
[A_{n+1}-X_{n+1}] &= [A_n-X_n]+[\mu(A_n)-\mu(X_n)]\delta-\sigma\sqrt{\delta}\,\epsilon_{n+1} \\
&\quad +\frac{\sigma^2}{\sigma^2+\tau^2}\Big([\mu(X_n)-\mu(A_n)]\delta+\sigma\sqrt{\delta}\,\epsilon_{n+1}+\tau\sqrt{\delta}\,\eta_{n+1}\Big)+\frac{\sigma\tau}{\sqrt{\sigma^2+\tau^2}}\sqrt{\delta}\,\zeta_{n+1} \\
&= [A_n-X_n]+\frac{2\sigma^2+\tau^2}{\sigma^2+\tau^2}[\mu(X_n)-\mu(A_n)]\delta \\
&\quad +\frac{\sigma^2\tau\sqrt{\delta}\eta_{n+1}-\sigma\tau^2\sqrt{\delta}\epsilon_{n+1}}{\sigma^2+\tau^2}+\frac{\sigma\tau}{\sqrt{\sigma^2+\tau^2}}\sqrt{\delta}\,\zeta_{n+1}.
\end{aligned}$$

A_n tracks X_n when the process $\{A_n-X_n, n=1,2,\dots\}$ is stable. This happens when $\mu(x)-\mu(y)$ is negative when x is sufficiently larger than y. For example, a stable autoregressive process with $\mu(x)=-ax$ gives a stable adapted filter process.

2.F Independent measurement error on a scale that gives a finite limiting amount of information about $X(t)$ from measurements on a unit time interval

We now consider the measurement model

$$\begin{aligned}
Y_{n+1} &= X_{n+1} + \frac{\tau}{\sqrt{\delta}}\,\eta_{n+1} \\
&= X_n + \mu(X_n)\,\delta + \sigma\sqrt{\delta}\,\epsilon_{n+1} + \frac{\tau}{\sqrt{\delta}}\eta_{n+1}, \qquad (2.56)
\end{aligned}$$

where $\{\epsilon_n, \eta_n\}$ is a collection of independent standard normal random variables. The conditional mean is now

$$\begin{aligned}
\mathbb{E}\big[X_{n+1}|Y_{n+1}, X_n\big]. &= X_n + \mu(X_n)\delta + \mathbb{E}\Big[\sigma\sqrt{\delta}\epsilon_{n+1}\,\Big|\,\sigma\sqrt{\delta}\epsilon_{n+1} + \frac{\tau}{\sqrt{\delta}}\eta_{n+1}\Big] \\
&= X_n + \mu(X_n)\delta + \frac{\sigma^2\delta}{\sigma^2\delta + \tau^2/\delta}\big(\sigma\sqrt{\delta}\epsilon_{n+1} + \tau\sqrt{\delta}\,\eta_{n+1}\big) \qquad (2.57)
\end{aligned}$$

Using (2.56) and (2.57) gives

$$\mathbb{E}\big[X_{n+1}|Y_{n+1}, X_n\big] = X_n + \mu(X_n)\delta + \frac{\sigma^2\delta^2}{\sigma^2\delta^2 + \tau^2}\Big(Y_{n+1} - X_n - \mu(X_n)\,\delta\Big).$$

In the limit as $\delta \to 0$, the contribution from the measurement is order δ^2 and is therefore negligible. Although the observation process is meaningfully informative about the latent process, the adapted simulation fails to track the latent process in this limit. Intuitively, this is because the adapted simulation is trying to track differences in the latent process, but for this model the signal to noise ratio for the difference in each interval of length δ tends to zero.

2.G Independent measurements of the latent process with measurement error on a scale that gives a useful adapted process as $\delta \to 0$

We now consider the measurement model

$$\begin{aligned} Y_{n+1} &= X_{n+1} + \tau\eta_{n+1} \\ &= X_n + \mu(X_n)\delta + \sigma\sqrt{\delta}\epsilon_{n+1} + \tau\eta_{n+1}. \end{aligned} \tag{2.58}$$

The conditional mean is now

$$\begin{aligned} \mathbb{E}\big[X_{n+1}|Y_{n+1}, X_n\big]. &= X_n + \mu(X_n)\,\delta + \mathbb{E}\big[\sigma\sqrt{\delta}\epsilon_{n+1}\,\big|\,\sigma\sqrt{\delta}\epsilon_{n+1} + \tau\eta_{n+1}\big] \\ &= X_n + \mu(X_n)\delta + \frac{\sigma^2\delta}{\sigma^2\delta + \tau^2}\big(\sigma\sqrt{\delta}\epsilon_{n+1} + \tau\eta_{n+1}\big) \end{aligned} \tag{2.59}$$

Using (2.58) and (2.59) gives

$$\begin{aligned} \mathbb{E}\big[X_{n+1}|Y_{n+1}, X_n\big] &= X_n + \mu(X_n)\delta + \frac{\sigma^2\delta}{\sigma^2\delta + \tau^2}\Big(Y_{n+1} - X_n - \mu(X_n)\delta\Big) \\ &= X_n + \mu(X_n)\delta + \frac{\sigma^2}{\tau^2}\delta\Big(Y_{n+1} - X_n - \mu(X_n)\,\delta\Big) + o(\delta) \end{aligned}$$

In the limit as $\delta \to 0$, the adapted simulation has a diffusive drift toward the value of the latent process.

For disease models, incidence data can arguably be considered as noisy measurements of the change of a state variable (number of susceptibles) that is not directly measured. This could correspond to a situation where the measurement error is on the same scale as the process noise (Appendix 2.E). Alternatively, we could think of weekly aggregated incidence as a noisy measurement of the infected class, in which case the measurement error could match the scaling in Appendix 2.G.

The model in Appendix 2.F is a cautionary tale, warning us against carrying out adapted simulation on short time intervals. An interpretation is that one should not carry out adapted simulation unless a reasonable amount of information has accrued. When each observation has low information, a particle filter may enable solution to the filtering problem without particle depletion. It is when the data are highly informative that the curse of dimensionality makes basic particle filters ineffective, opening up demand for alternative methods.

We are now in a better position to understand why it may be appropriate to keep many particle representations at intermediate timesteps while resampling down to a single representative at each observation time, as ABF and ABF-IR do. We have seen that adaptive simulation can fail when observations occur frequently. Resampling down to a single particle too often can lose the ability for the adapted process to track the latent process. This implies that adapted simulation should not be relied upon more than necessary to ameliorate the curse of dimensionality: once proper importance sampling for filtering problem becomes tractable in a sufficiently small spatiotemporal neighborhood, one should maintain weighted particles on this spatiotemporal scale rather than resorting to adapted simulation.

2.H Bagged filters for functions of the latent states

The theory and methodology in the main article focused on filtering for likelihood estimation. Here, we describe extensions to other filtering problems. Let $\{\phi_k : \boldsymbol{X}^U \to \mathbb{R},\ k \text{ in } 1\!:\!K\}$ be a collection of functions where ϕ_k depends only on a subset of units $\Phi_k \subset 1\!:\!U$. We suppose that there exist neighborhoods

$$B'_{k,n} \subset A'_n = (1\!:\!U) \times (0\!:\!n) \tag{2.60}$$

such that $\phi_k(\boldsymbol{X}_n)$ is approximately independent of $\{Y_{u,n} : (u,n) \in B'^c_{k,n}\}$ given $\{Y_{u,n} : (u,n) \in B'_{k,n}\}$, where $B'^c_{k,n}$ is a complement in A'_n. Unlike the sets $A_{u,n}$ and $B_{u,n}$ defined for likelihood estimation, the sets A'_n and $B'_{k,n}$ can include any locations at time n. We consider Monte Carlo estimation of

$$\overline{\phi}_{k,n} = \mathbb{E}\big[\phi_k(\boldsymbol{X}_n)\big|\boldsymbol{Y}_{1:n}\big], \quad k \text{ in } 1\!:\!K. \tag{2.61}$$

For example, if we set $\phi_u(\boldsymbol{x}_n) = x_{u,n}$ and $K = U$, the collection of quantities $\{\overline{\phi}_{u,n}\}$ estimated in (2.61) corresponds to a vector of filter means. Setting $\phi_k(\boldsymbol{x}_n) = x_{u_k,n}\, x_{\tilde{u}_k,n}$ enables calculation of a collection of filter covariances between units u_k and $\tilde{u}_k$ for k in $1\!:\!K$.

We consider bagged filtering approaches to estimation of $\{\overline{\phi}_{k,n}, k \text{ in } 1:K\}$. Although each $\overline{\phi}_{k,n}$ is required to have only local dependence, some global quantities such as filter means and their variances across all units, can be expressed in terms of collections of such quantities. For bagged filtering to operate successfully, the neighborhoods $\{B'_{k,n}\}$ should not be large. Since $B'_{k,n}$ will typically be larger than $\{\Phi_k\}$, this rules out estimation of filtered quantities that cannot be adequately represented by a collection of localized filtering calculations. We now present variants of the pseudocode in the main text, targeted at estimation of $\{\overline{\phi}_{k,n}, k \text{ in } 1\!:\!K\}$. We do not prove theorems about these algorithms, but we conjecture from their similarity to the algorithms in the main text that comparable theoretical results should exist. Table 2.3 lists the inputs, outputs and ranges of the implicit loops for the following algorithms.

Latent state estimation via bagged filters.

input:
collection of functions, ϕ_k
neighborhoods, $B'_{k,n}$
simulator for $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)$ and $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{x}_{n-1})$
evaluator for $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \,|\, x_{u,n})$
number of replicates, $\mathcal{I}$
data, $\boldsymbol{y}^*_{1:N}$
ABF and ABF-IR: particles per replicate, J
ABF-IR: number of intermediate timesteps, S
ABF-IR: measurement variance parameterizations, $\overleftarrow{\mathrm{v}}_{u,n}$ and $\overrightarrow{\mathrm{v}}_{u,n}$
ABF-IR: approximate process and observation mean functions, $\boldsymbol{\mu}$ and $h_{u,n}$
output:
filter estimate, $\overline{\phi}^{\,\mathrm{MC}}_{k,n}$
implicit loops:
u in $1\!:\!U$, n in $1\!:\!N$, i in $1\!:\!\mathcal{I}$, j in $1\!:\!J$, k in $1\!:\!K$

Table 2.3 – Notation for bagged filter for latent state estimation: inputs, outputs and implicit loops.

UBF. Unadapted bagged filter for latent state estimation.

```
Simulate $\boldsymbol{X}_{0:N,i} \sim f_{\boldsymbol{X}_{0:N}}(\boldsymbol{x}_{0:N})$
Measurement weights, $w^{M}_{u,n,i} = f_{Y_{u,n}|X_{u,n}}(y^{*}_{u,n} \,|\, X_{u,n,i})$
Filtering weights, $w^{F}_{k,n,i} = \prod_{(\tilde{u},\tilde{n})\in B'_{k,n}} w^{M}_{\tilde{u},\tilde{n},i}$
$\overline{\phi}^{\,\mathrm{MC}}_{k,n} = \dfrac{\sum_{i=1}^{\mathcal{I}} \phi_k(\boldsymbol{X}_{n,i})\, w^{F}_{k,n,i}}{\sum_{i=1}^{\mathcal{I}} w^{F}_{k,n,i}}$
```

ABF. Adapted bagged filter for latent state estimation.

```
Initialize adapted simulation: $\boldsymbol{X}^{\mathrm{A}}_{0,i} \sim f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)$
For $n$ in $1\!:\!N$
    Proposals: $\boldsymbol{X}^{\mathrm{P}}_{n,i,j} \sim f_{\boldsymbol{X}_n|X_{1:U,n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{X}^{\mathrm{A}}_{n-1,i})$
    Measurement weights: $w^{M}_{u,n,i,j} = f_{Y_{u,n}|X_{u,n}}(y^{*}_{u,n} \,|\, X^{\mathrm{P}}_{u,n,i,j})$
    Adapted resampling weights: $w^{\mathrm{A}}_{n,i,j} = \prod_{u=1}^{U} w^{M}_{u,n,i,j}$
    Resampling: $\mathbb{P}[r(i)=a] = w^{\mathrm{A}}_{n,i,a}\Big(\sum_{q=1}^{J} w^{\mathrm{A}}_{n,i,q}\Big)^{-1}$
    $\boldsymbol{X}^{\mathrm{A}}_{n,i} = \boldsymbol{X}^{\mathrm{P}}_{n,i,r(i)}$
    $w^{F}_{k,n,i,j} = \prod_{\tilde{n}=1}^{n-1}\Big[\frac{1}{J}\sum_{q=1}^{J}\prod_{\tilde{u}:(\tilde{u},\tilde{n})\in B'_{k,n}} w^{M}_{\tilde{u},\tilde{n},i,q}\Big]\prod_{\tilde{u}:(\tilde{u},n)\in B'_{k,n}} w^{M}_{\tilde{u},n,i,j}$
End for
$\overline{\phi}^{\,\mathrm{MC}}_{k,n} = \dfrac{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} \phi_k(\boldsymbol{X}^{\mathrm{P}}_{n,i,j})\, w^{F}_{k,n,i,j}}{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^{F}_{k,n,i,j}}$
```

ABF-IR. Adapted bagged filter with intermediate resampling for latent state estimation.

```
Initialize adapted simulation: $\boldsymbol{X}^{\mathrm{A}}_{0,i} \sim f_{\boldsymbol{X}_0}(\boldsymbol{x}_0)$
For $n$ in $1\!:\!N$
    Guide simulations: $\boldsymbol{X}^{G}_{n,i,j} \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}\big(\boldsymbol{x}_n \,|\, \boldsymbol{X}^{\mathrm{A}}_{n-1,i}\big)$
    Guide sample variance: $V_{u,n,i} = \mathrm{Var}\big\{h_{u,n}\big(X^{G}_{u,n,i,j}\big), j \text{ in } 1\!:\!J\big\}$
    $g^{\mathrm{R}}_{n,0,i,j} = 1$ and $\boldsymbol{X}^{\mathrm{IR}}_{n,0,i,j} = \boldsymbol{X}^{\mathrm{A}}_{n-1,i}$
    For $s$ in $1\!:\!S$
        Intermediate proposals: $\boldsymbol{X}^{\mathrm{IP}}_{n,s,i,j} \sim f_{\boldsymbol{X}_{n,s}|\boldsymbol{X}_{n,s-1}}\big(\,\cdot\,|\boldsymbol{X}^{\mathrm{IR}}_{n,s-1,i,j}\big)$
        $\boldsymbol{\mu}^{\mathrm{IP}}_{n,s,i,j} = \boldsymbol{\mu}\big(\boldsymbol{X}^{\mathrm{IP}}_{n,s,i,j}, t_{n,s}, t_n\big)$
        $V^{\mathrm{meas}}_{u,n,s,i,j} = \overrightarrow{\mathrm{v}}_u(\theta, \mu^{\mathrm{IP}}_{u,n,s,i,j})$
        $V^{\mathrm{proc}}_{u,n,s,i} = V_{u,n,i}\,\big(t_n - t_{n,s}\big)\Big/\big(t_n - t_{n,0}\big)$
        $\theta_{u,n,s,i,j} = \overleftarrow{\mathrm{v}}_u\big(V^{\mathrm{meas}}_{u,n,s,i,j} + V^{\mathrm{proc}}_{u,n,s,i},\, \mu^{\mathrm{IP}}_{u,n,s,i,j}\big)$
        $g_{n,s,i,j} = \prod_{u=1}^{U} f_{Y_{u,n}|X_{u,n}}\big(y^*_{u,n} \,|\, \mu^{\mathrm{IP}}_{u,n,s,i,j}\,; \theta_{u,n,s,i,j}\big)$
        Guide weights: $w^{G}_{n,s,i,j} = g_{n,s,i,j}\big/g^{\mathrm{R}}_{n,s-1,i,j}$
        Resampling: $\mathbb{P}\big[r(i,j) = a\big] = w^{G}_{n,s,i,a}\Big(\sum_{q=1}^{J} w^{G}_{n,s,i,q}\Big)^{-1}$
        $\boldsymbol{X}^{\mathrm{IR}}_{n,s,i,j} = \boldsymbol{X}^{\mathrm{IP}}_{n,s,i,r(i,j)}$ and $g^{\mathrm{R}}_{n,s,i,j} = g_{n,s,i,r(i,j)}$
    End For
    Set $\boldsymbol{X}^{\mathrm{A}}_{n,i} = \boldsymbol{X}^{\mathrm{IR}}_{n,S,i,1}$
    Measurement weights: $w^{M}_{u,n,i,j} = f_{Y_{u,n}|X_{u,n}}\big(y^*_{u,n} \,|\, X^{G}_{u,n,i,j}\big)$
    $w^{F}_{k,n,i,j} = \prod_{\tilde{n}=1}^{n-1}\Big[\frac{1}{J}\sum_{q=1}^{J}\prod_{\tilde{u}:(\tilde{u},\tilde{n})\in B'_{k,n}} w^{M}_{\tilde{u},\tilde{n},i,q}\Big]\prod_{\tilde{u}:(\tilde{u},n)\in B'_{k,n}} w^{M}_{\tilde{u},n,i,j}$
End for
$\overline{\phi}^{\,\mathrm{MC}}_{k,n} = \dfrac{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J}\phi_k(\boldsymbol{X}^{\mathrm{P}}_{n,i,j})\,w^{F}_{k,n,i,j}}{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^{F}_{k,n,i,j}}$
```

Chapter 3

Partially Observed Markov Processes with Spatial Structure via the R Package spatPomp

3.1 Introduction

A partially observed Markov process (POMP) model consists of incomplete and noisy measurements of a latent Markov process. A POMP model in which the latent process has spatial as well as temporal structure is called a spatiotemporal POMP or SpatPOMP. Many biological, social and physical systems have the spatiotemporal structure, dynamic stochasticity and imperfect observability that characterize SpatPOMP models. The spatial structure of SpatPOMPs adds complexity to the problems of likelihood estimation, parameter inference and model selection for nonlinear and non-Gaussian systems. The objective of the **spatPomp** package is to facilitate model development and data analysis in the context of the general class of SpatPOMP models, enabling scientists to separate the scientific task of model development from the statistical task of providing inference tools. Thus, **spatPomp** brings together general purpose methods for carrying out Monte Carlo statistical inference for such systems. More generally, **spatPomp** provides an abstract representation for specifying SpatPOMP models. This ensures that SpatPOMP models formulated with the package can be investigated using a range of methods, and that new methods can be readily tested on a range of models. In its current manifestation, **spatPomp** is appropriate for data analysis with a moderate number of spatial units (up to around 100 spatial units) having nonlinear and non-Gaussian dynamics. In particular, **spatPomp** is not targeted at very large spatiotemporal systems like those that are common in geophysical data assimilation. The data assimilation research testbed (Anderson et al., 2009, DART) is a notable software facility that has contributed to projects that study such large systems. Spatiotemporal systems with Gaussian dynamics can be investigated with **spatPomp**, but a variety of alternative methods and software are available in this case (Wikle et al., 2019; Sigrist et al., 2015; Cappello et al., 2020).

The **spatPomp** package builds on the **pomp** package described by King et al. (2016). Mathematically, a SpatPOMP model is also a POMP model, and this property is reflected in the object-oriented design of **spatPomp**: The package is implemented using S4 classes (Chambers, 1998; Genolini, 2008; Wickham, 2019) and the basic class '`spatPomp`' extends the class '`pomp`' provided by **pomp**. Among other things, this allows us to test new methods against extensively tested methods in low-dimensional settings, use existing convenience functions, and apply methods for POMP models on SpatPOMP models. However, standard Monte Carlo statistical inference methods for nonlinear POMP models break down with increasing spatial dimension (Bengtsson et al., 2008), a manifestation of the *curse of dimensionality.* Therefore, effective inference approaches must, in practice, take advantage of the special structure of SpatPOMP models. Figure 3.1 illustrates the use case of the

spatPomp package relative to the **pomp** package and methods that use Gaussian approximations to target models with massive dimensionality. The difficulty of statistical inference for a dynamic model can be thought of as a combination of its nonlinearity and its dimensionality, so methods for nonlinear dynamics of small dimensions can work well for linear and Gaussian problems of relatively high dimensions. This means that the boundaries of the methods and packages in Figure 3.1 can extend beyond the edges shown in the figure. Nevertheless, it is useful to situate models in this nonlinearity-dimensionality problem space so that candidate methods and software packages can become clearer.

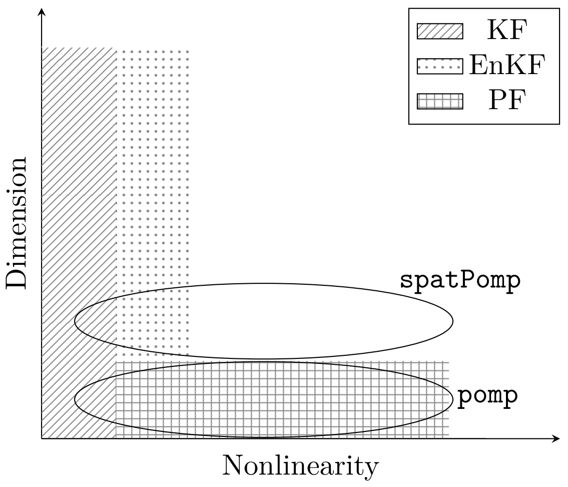


Figure 3.1 – Diagram illustrating the use case for the **spatPomp** package. For statistical inference of models that are approximately linear and Gaussian, the Kalman Filter (KF) is an appropriate method. If the nonlinearity in the problem increases moderately but the dimension of the problem is very large (e.g. geophysical models), the ensemble Kalman Filter (EnKF) and similar methods are useful. In low-dimensional but very nonlinear settings, the particle filter (PF) and related methods are useful and the **pomp** package targets such problems. The **spatPomp** package and the methods implemented in it are intended for statistical inference for nonlinear models that are of moderate dimension. The nonlinearity in these models (e.g. epidemiological models) is too large for Gaussian approximations and the dimensionality is large enough to cause the particle filter to be unstable.

A SpatPOMP model is characterized by the transition density for the latent Markov process and unit-specific measurement densities[1]. Once these elements are specified, calculating and simulating

[1]We use the term "density" in this article to encompass both the continuous and discrete cases. Thus, when latent

from all joint and conditional densities are well defined operations. However, different statistical methods vary in the operations they require. Some methods require only simulation from the transition density whereas others require evaluation of this density. Some methods avoid working with the model directly, replacing it by an approximation, such as a linearization. For a given model, some operations may be considerably easier to implement and so it is useful to classify inference methods according to the operations on which they depend. In particular, an algorithm is said to be *plug-and-play* if it utilizes simulation of the latent process but not evaluation of transition densities (Bretó et al., 2009; He et al., 2010). The arguments for and against plug-and-play methodology for SpatPOMP models are essentially the same as for POMP models (He et al., 2010; King et al., 2016). Simulators are relatively easy to implement for most SpatPOMP models; plug-and-play methodology facilitates the investigation of a variety of models that may be scientifically interesting but mathematically inconvenient. On the other hand, approaches that leverage explicit transition densities are sometimes more computationally efficient than those that rely on Monte Carlo methods. Nevertheless, the utility of plug-and-play methods has been amply demonstrated in scientific applications. In particular, plug-and-play methods implemented using **pomp** have proved capable for state-of-the-art inference problems for POMP models (e.g., King et al., 2008; Bhadra et al., 2011; Shrestha et al., 2011, 2013; Earn et al., 2012; Roy et al., 2013; Blackwood et al., 2013a,b; He et al., 2013; Bretó, 2014; Blake et al., 2014; Martinez-Bakker et al., 2015; Bakker et al., 2016; Becker et al., 2016; Buhnerkempe et al., 2017; Ranjeva et al., 2017; Marino et al., 2019; Pons-Salort and Grassly, 2018; Becker et al., 2019; Kain et al., 2020). Although the **spatPomp** package provides a general environment for methods with and without the plug-and-play

variables or measured quantities are discrete, one can replace "probability density function" with "probability mass function".

property, development of the package to date has emphasized plug-and-play methods.

The remainder of this paper is organized as follows. Section 3.2 defines mathematical notation for SpatPOMP models and relates this to their representation as objects of class '`spatPomp`' in the **spatPomp** package. Section 3.3 introduces simulation and several spatiotemporal filtering methods currently implemented in **spatPomp**. Section 3.4 introduces some parameter estimation algorithms currently implemented in **spatPomp** which build upon these simulation and filtering techniques. Section 3.5 constructs a simple linear Gaussian SpatPOMP model and uses this example to illustrate the statistical methodology. Section 3.6 discusses the construction of spatially structured compartment models for population dynamics, in the context of coupled measles dynamics in UK cities; this demonstrates the kind of nonlinear stochastic system primarily motivating the development of **spatPomp**. Finally, Section 3.7 discusses extensions and applications of **spatPomp**.

3.2 SpatPOMP models and their representation in spatPomp

We now set up notation for SpatPOMP models. This notation is similar to that of POMP models (King et al., 2016), but we provide all the details here for completeness. A visually intuitive schematic illustrating SpatPOMPs is given in Figure 3.2. The notation allows us to eventually define a SpatPOMP model in terms of its three main components: a model for one-step transitions of the latent states, a model for the measurements at an observation time conditional on the latent states at that time, and an initializer for the latent state process. Suppose there are U units labeled $1\!:\!U = \{1, 2, \dots, U\}$. Let $\mathbb{T}$ be the set of times at which the latent dynamic process is defined. The SpatPOMP framework and the **spatPomp** package permit the timescale $\mathbb{T}$ to be a discrete or continuous subset of $\mathbb{R}$. In either case, $\mathbb{T}$ must contain a set of N observation times, $t_{1:N} = \{t_n : n =$

$1, \ldots, N\}$, where $t_1 < t_2 < \cdots < t_N$. In the examples below, we take $\mathbb{T} = [t_0, t_N]$, with $t_0 \le t_1$ being the time at which we begin modeling the dynamics of the latent Markov process. The unobserved, latent Markov process is written as $\{\boldsymbol{X}(t\,;\theta) : t \in \mathbb{T}\}$ where $\boldsymbol{X}(t\,;\theta) = (X_1(t\,;\theta), \ldots, X_U(t\,;\theta))$, with $X_u(t\,;\theta)$ taking values in some space $\mathbb{R}^{D_X}$ and θ a D_θ-dimensional real-valued parameter which we write as $\theta = \theta_{1:D_\theta} \in \mathbb{R}^{D_\theta}$. For some purposes it is adequate to consider the latent process only at the finite collection of times at which it is observed. We write $\boldsymbol{X}_n = \boldsymbol{X}(t_n\,;\theta)$, noting that we sometimes choose to suppress the dependence on θ. We can also write $\boldsymbol{X}_n = (X_{1,n}, \ldots, X_{U,n}) = X_{1:U,n}$. The initial value $\boldsymbol{X}_0 = \boldsymbol{X}(t_0\,;\theta)$ may be stochastic or deterministic. The observable process $\{\boldsymbol{Y}_n = Y_{1:U,n}, n \in 1\!:\!N\}$ takes values in $\mathbb{R}^U$.

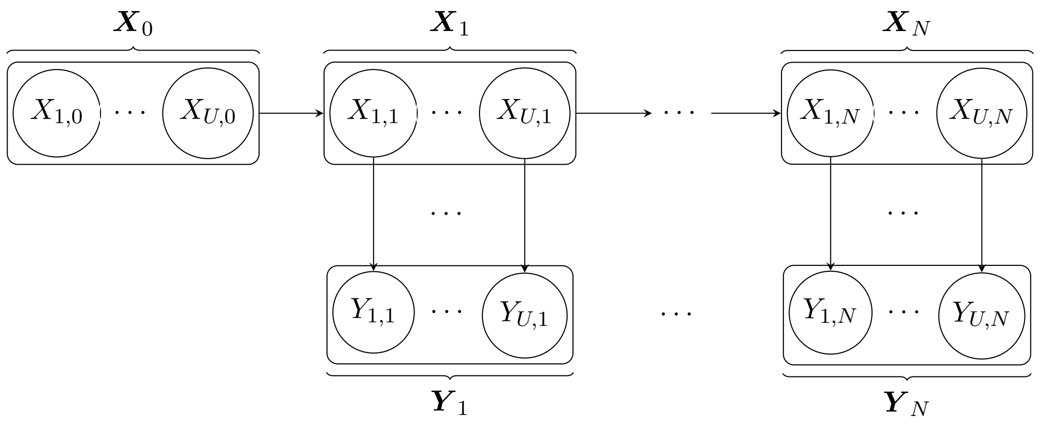


Figure 3.2 – Diagram illustrating spatially coupled partially observed Markov process (SpatPOMP) models. The latent dynamic model is $\{\boldsymbol{X}(t;\theta) : t \in \mathbb{T}\}$. During observation times $t_{1:N} = \{t_n : n = 1, \ldots, N\}$, the values of the latent process are denoted $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$. The partial and noisy observations at these times are denoted $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_N$. A SpatPOMP model specifies that $\boldsymbol{X}(t)$ is itself a collection of latent states from U spatially coupled dynamics that each produce observations at the observation times. Ultimately, defining a SpatPOMP model is equivalent to defining three models: a model for the one-step transition of the latent process, a model for each spatial unit's measurements at an observation time conditional on the latent states for that unit at that time, and a model for the initial latent state, $\boldsymbol{X}_0$.

Observations are modeled as conditionally independent given the latent process. This conditional independence of measurements applies over both space and time, so $Y_{u,n}$ is conditionally independent of $\{Y_{\tilde{u},\tilde{n}}, (\tilde{u},\tilde{n}) \neq (u,n)\}$ given $X_{u,n}$. We suppose the existence of a joint density $f_{\boldsymbol{X}_{0:N},\boldsymbol{Y}_{1:N}}$ of $X_{1:U,0:N}$ and $Y_{1:U,1:N}$ with respect to some reference measure. We use the same subscript notation to denote other marginal and conditional densities. The data, $y^*_{1:U,1:N} = (\boldsymbol{y}^*_1, \ldots, \boldsymbol{y}^*_N)$, are modeled as a realization of this observation process. Spatially or temporally dependent measurement errors can be modeled by adding suitable latent variables.

The SpatPOMP structure defined above is equivalent to the following factorization of the joint density in terms of the initial density, $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0;\theta)$, together with the conditional transition probability density,

$f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{x}_{n-1}\,;\theta)$, and the unit-specific measurement densities, $f_{Y_{u,n}|X_{u,n}}(y_{u,n}|x_{u,n}\,;\theta)$ for $1 \leq n \leq N$, $1 \leq u \leq U$:

$$f_{\boldsymbol{X}_{0:N},\boldsymbol{Y}_{1:N}}(\boldsymbol{x}_{0:N},\boldsymbol{y}_{1:N};\theta) = f_{\boldsymbol{X}_0}(\boldsymbol{x}_0;\theta)\prod_{n=1}^{N} f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n|\boldsymbol{x}_{n-1};\theta)\prod_{u=1}^{U} f_{Y_{u,n}|X_{u,n}}(y_{u,n}|x_{u,n}\,;\theta).$$

This formalism allows the transition density, $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}$, and unit-specific measurement density, $f_{Y_{u,n}|X_{u,n}}$, to depend on n and u, so we allow for the possibility of temporally and spatially inhomogeneous models.

3.2.1 Implementation of SpatPOMP models

A SpatPOMP model is represented in **spatPomp** by an `S4` object of class '`spatPomp`'. Slots in this object encode the components of the SpatPOMP model, and can be filled or changed using the constructor function `spatPomp()` and various other convenience functions. Methods for the class '`spatPomp`' use these components to carry out computations on the model. Table 3.1 gives the mathematical notation corresponding to the elementary methods that can be executed on a

Method	Argument to `spatPomp()`	Mathematical terminology
`dunit_measure`	`dunit_measure`	Evaluate $f_{Y_{u,n}\vert X_{u,n}}(y_{u,n}\,\vert\,x_{u,n}\,;\theta)$
`runit_measure`	`runit_measure`	Simulate from $f_{Y_{u,n}\vert X_{u,n}}(y_{u,n}\,\vert\,x_{u,n}\,;\theta)$
`eunit_measure`	`eunit_measure`	Evaluate $\mathrm{e}_{u,n}(x,\theta)=\mathbb{E}[Y_{u,n}\,\vert\,X_{u,n}=x\,;\theta]$
`vunit_measure`	`vunit_measure`	Evaluate $\mathrm{v}_{u,n}(x,\theta)=\mathrm{Var}[Y_{u,n}\,\vert\,X_{u,n}=x\,;\theta]$
`munit_measure`	`munit_measure`	$\mathrm{m}_{u,n}(x,V,\theta)=\boldsymbol{\psi}$ solves $\mathrm{v}_{u,n}(x,\boldsymbol{\psi})=V$, $\mathrm{e}_{u,n}(x,\boldsymbol{\psi})=\mathrm{e}_{u,n}(x,\theta)$
`rprocess`	`rprocess`	Simulate from $f_{\boldsymbol{X}_n\vert\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n\,\vert\,\boldsymbol{x}_{n-1}\,;\theta)$
`dprocess`	`dprocess`	Evaluate $f_{\boldsymbol{X}_n\vert\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n\,\vert\,\boldsymbol{x}_{n-1}\,;\theta)$
`rmeasure`	`rmeasure`	Simulate from $f_{\boldsymbol{Y}_n\vert\boldsymbol{X}_n}(\boldsymbol{y}_n\,\vert\,\boldsymbol{x}_n\,;\theta)$
`dmeasure`	`dmeasure`	Evaluate $f_{\boldsymbol{Y}_n\vert\boldsymbol{X}_n}(\boldsymbol{y}_n\,\vert\,\boldsymbol{x}_n\,;\theta)$
`rprior`	`rprior`	Simulate from the prior distribution $\pi(\theta)$
`dprior`	`dprior`	Evaluate the prior density $\pi(\theta)$
`rinit`	`rinit`	Simulate from $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$
`timezero`	`t0`	t_0
`time`	`times`	$t_{1:N}$
`obs`	`data`	$\boldsymbol{y}^*_{1:N}$
`states`	—	$\boldsymbol{x}_{0:N}$
`coef`	`params`	θ

Table 3.1 – Model component methods for class ‘`spatPomp`’ objects. A translation into mathematical notation for SpatPOMP models is also included. For example, the `rprocess` method is set using the `rprocess` argument to the `spatPomp` constructor function.

class ‘`spatPomp`’ object. Class ‘`spatPomp`’ inherits from the class ‘`pomp`’ defined by the **pomp** package. One of the main ways in which **spatPomp** extends **pomp** is the addition of unit-level specifications of the measurement model. This reflects the modeling assumption that measurements are carried out independently in both space and time, conditional on the current value of the latent process which is known as the *state* of the dynamic system. There are five unit-level functionalities of class ‘`spatPomp`’ objects: `dunit_measure`, `runit_measure`, `eunit_measure`, `vunit_measure` and `munit_measure`. Each functionality corresponds to an S4 method. The set of instructions performed by each method are supplied by the user via an argument to the `spatPomp()` constructor function of the same name. (See Table 3.1 for details).

Only functionalities that are required to run an algorithm of interest need to be supplied in advance. `dunit_measure` evaluates the probability density of the measurement of a spatial unit given its latent state vector, whereas `runit_measure` simulates from this conditional distribution. Given the latent state, `eunit_measure` and `vunit_measure` give the expectation and variance of the measurement, respectively. They are used by the ensemble Kalman filter (EnKF, Section 3.3.3) and iterated EnKF (Section 3.4.2). `munit_measure` returns a parameter vector corresponding to given moments (mean and variance), used by one of the options for a guided particle filter (Section 3.3.1).

3.2.2 Initial conditions

Specification of the initial condition $\boldsymbol{X}_0 = \boldsymbol{X}(t_0;\theta)$ of a SpatPOMP model is similar to that of a POMP model, and is carried out using the `rinit` argument of the `spatPomp()` constructor function. The initial distribution $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$ may sometimes be a known property of the system but in general it must be inferred. If the transition density for the dynamic model, $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n\,|\,\boldsymbol{x}_{n-1}\,;\theta)$, does not depend on time and possesses a unique stationary distribution, it may be natural to set $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$ to be this stationary distribution. When no clear scientifically motivated choice of $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$ exists, one can treat $\boldsymbol{X}_0$ as a component of the parameter set to be estimated. In this case, $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$ concentrates at a point which depends on θ.

3.2.3 Covariates

Scientifically, one may be interested in the impact of a vector-valued covariate process $\{\mathcal{Z}(t)\}$ on the latent dynamic system. For instance, the transition density, $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}$, and the measurement density, $f_{\boldsymbol{Y}_n|\boldsymbol{X}_n}$, can depend on this observed process. The covariate process might be shared between units, or might take unit-specific values. **spatPomp** allows modeling and inference for spatiotemporal processes that depend on an observed covariate process. If such a covariate exists the user can supply a class ‘`data.frame`’ object to the `covar` argument of the `spatPomp()` constructor function. This `data.frame` will have a column for time, spatial unit, and each of the covariates. If any of the variables in the covariates `data.frame` is common among all units the user must supply the variable names as class ‘`character`’ vectors to the `shared_covarnames` argument of the `spatPomp()` constructor function. Otherwise, all covariates are assumed to be unit-specific, meaning that they generally take on different values for each unit. **spatPomp** manages the task of presenting interpolated values of the covariates to the elementary model functions at the time they are called. An example implementing a SpatPOMP model with covariates is presented in Section 3.6.

3.2.4 Naming units and components of the state

One of the requirements for creating a new class ‘`spatPomp`’ object is a class ‘`data.frame`’ object containing observations for each spatial unit at each time. This long-format `data.frame` is supplied to the `data` argument of the `spatPomp()` constructor function. The package does not presuppose data at all units at all observation times. Missing data in some or all of the observations for any given observation times are preserved, which allows the user to describe a measurement model that handles missingness. The name of the `data` column containing observation times is supplied to the `times` argument; the name of the column containing the unit names is supplied to the `units` argument. The `t0` argument, for which the user supplies the initial time of the dynamics in the time unit of the measurements, is the last of the four required arguments for the `spatPomp()` constructor. The resulting class ‘`spatPomp`’ object stores the unit names as a vector in a slot called `unit_names`. Internally, only the positions of the names in the `unit_names` vector are used to keep track of units. In other words, the units are labeled 1 through `U` just as in our mathematical notation, and these labels can be used to construct measurement or process models that differ between units. The text labels in `unit_names` are re-attached for user-readable outputs like simulation plots and class '`data.frame`' outputs.

Another important argument to the `spatPomp()` constructor is `unit_statenames`. This argument expects a class '`character`' vector of length D_X containing names of the components of any unit's latent state at a given time. For example, to implement a SpatPOMP model studying the population dynamics between frogs and dragonflies in neighboring spatial units, the user can provide `unit_statenames = c('F','D')`. The package then expects that unit-level model components will use `'F'` and `'D'` in their instructions.

Other name vectors, `unit_obsnames` and `unit_covarnames`, are used internally to keep track of data and covariates that have corresponding values for each unit. These need not be supplied in the `spatPomp()` constructor, however, since the data and covariate `data.frame` objects provided by the user implicitly supply them.

3.2.5 Specifying model components using C snippets

The **spatPomp** package utilizes the C snippet facility in **pomp** which allows users to specify the model components in Table 3.1 using inline C code. The package is therefore able to perform computationally expensive calculations in C while outputting results in higher-level R objects. The code below illustrates the creation of a C snippet unit measurement density using the `Csnippet()` function.

```
R> example_dunit_measure <- Csnippet("
+    // define measurement standard deviation for first unit
+    double sd0 = sd*1.5;
+    // Quantities u, Y, X, sd and lik are available in the context
+     // in which this snippet is executed.
+     if (u==0) lik = dnorm(Y, X, sd0, give_log);
+     else lik = dnorm(Y, X, sd, give_log);
+     "
+   )
```

The example C snippet above illustrates a key difference in the practical use of the five unit-level model components in Table 3.1 compared to the components inherited from **pomp** that access the entire state and measurement vectors. All the filtering and inference algorithms in **spatPomp** assume the conditional independence of the spatial unit measurements given the state corresponding to the unit.

$$f_{\boldsymbol{Y}_n|\boldsymbol{X}_n}(\boldsymbol{y}_n \,|\, \boldsymbol{x}_n \,; \theta) = \prod_{u=1}^{U} f_{Y_{u,n}|X_{u,n}}(y_{u,n} \,|\, x_{u,n} \,; \theta)$$

When we specify the unit-level model components we can thus assume that the segments of the measurement and state vectors for the current unit are passed in during the execution of the unit-level model component. This allows the user to declare the unit-level model components by using the `unit_statenames` and `unit_obsnames` without having to specify the names of the relevant components of the latent state and observation. For instance, the example C snippet uses `X` instead of `X1`, `X2`, etc. Similarly, it uses `Y` instead of `Y1`, `Y2`, etc.

The variable `u`, which takes a value between `0` and `U-1`, is passed in to each unit-level model component. This allows the user to specify heterogeneity in the unit-level model components across units. Since C uses zero-based numbering, a user interested in introducing model heterogeneity for a unit must find the position of the unit in the `unit_names` slot and subtract one to get the corresponding value of `u`. The user can then use standard conditioning logic to specify the heterogeneity. For instance, when the `example_dunit_measure` C snippet above is executed for spatial unit 1, the `u` is passed in as `0`, `Y` will have the value of the measurement from unit 1 (or `Y1`) and `X` will have the value of the latent state in unit 1 (or `X1`). This example `C` snippet is coded so that the first unit has a measurement error inflated relative to that of the other units. As in **pomp**, the `give_log` variable is a logical flag indicating whether the desired output is on log scale: in this example, it is passed to the `log` argument of `dnorm`. Setting `give_log=TRUE` allows the calling algorithm to maintain accuracy when measurement densities become very small; the same convention is followed by all base `R` probability distribution functions.

Not all of the model components need to be supplied for any specific computation. In particular, plug-and-play methodology by definition never uses `dprocess`. An empty `dprocess` slot in a class '`spatPomp`' object is therefore acceptable unless a non-plug-and-play algorithm is attempted. In the package, the data and corresponding measurement times and units are considered necessary parts of a class '`spatPomp`' object whilst specific values of the parameters and latent states are not.

3.2.6 Examples included in the package

The construction of a new class '`spatPomp`' object is illustrated in Section 3.6. To provide some examples of class '`spatPomp`' objects, the **spatPomp** package includes functions `bm()`, `lorenz()` and `measles()`. These functions create class '`spatPomp`' objects with user-specified dimensions for a correlated Brownian motion model, the Lorenz-96 atmospheric model (Lorenz, 1996), and a spatiotemporal measles SEIR model, respectively. These examples can be used to better understand the components of class '`spatPomp`' objects as well as to test filtering and inference algorithms for future development.

For instance, we can create four correlated Brownian motions each with ten time steps as follows. The correlation structure and other model details are discussed in Section 3.5.

```
R> U <- 4; N <- 10
R> bm4 <- bm(U = U, N = N)
```

The above code results in the creation of the `bm4` object of class '`spatPomp`' with simulated data. This is done by bringing together pre-specified `C` snippets and adapting them to a four-dimensional process. One can inspect many aspects of `bm4`, some of which are listed in Table 3.1, by using the corresponding accessor functions:

```
R> obs(bm4)
R> unit_names(bm4)
R> states(bm4)
R> as.data.frame(bm4)
R> plot(bm4)
R> timezero(bm4)
R> time(bm4)
R> coef(bm4)
R> rinit(bm4)
```

The `measles()` example is described in Section 3.6 to demonstrate user-specified model construction.

3.3 Simulation and filtering: Elementary SpatPOMP methods

Once the user has encoded one or more SpatPOMP models as objects of class '`spatPomp`', the package provides a variety of algorithms to assist with data analysis. Methods can be divided into two categories: *elementary* operations that investigate a model with a fixed set of parameter values, and *inference* operations that estimate parameters. This section considers the first of these categories.

A basic operation on a SpatPOMP model is to simulate a stochastic realization of the latent process and the resulting data. This requires specifications of `rprocess` and `rmeasure`. Applying the `simulate` function to an object of class '`spatPomp`' by default returns another object of class '`spatPomp`', within which the data $\boldsymbol{y}^*_{1:N}$ have been replaced by a stochastic realization of $\boldsymbol{Y}_{1:N}$. The corresponding realization of $\boldsymbol{X}_{0:N}$ is accessible via the `states` slot, and the `params` slot is filled with the value of θ used in the simulation. Optionally, `simulate` can be made to return a class '`data.frame`' object by supplying the argument `format='data.frame'` in the call to `simulate`. Section 3.6 provides an example of constructing a class '`spatPomp`' object and simulating from it.

Evaluating the conditional distribution of latent process variables given currently available data is an elementary operation called *filtering*. Filtering also provides an evaluation of the likelihood function for a fixed parameter vector. The curse of dimensionality associated with spatiotemporal models can make filtering for SpatPOMP models computationally challenging. A widely used time-series filtering technique is the particle filter, available as `pfilter` in the **pomp** package. However, most particle filter algorithms scale poorly with dimension (Bengtsson et al., 2008; Snyder et al., 2015). Thus, in the spatiotemporal context, successful particle filtering requires state-of-the-art algorithms. Currently the **spatPomp** package contains implementations of five such algorithms, four of which are described below. Spatiotemporal data analysis using mechanistic models is a nascent topic, and future methodological developments are anticipated. Since the mission of **spatPomp** is to be a home for such analyses, the package developers welcome contributions and collaborations to further expand the functionality of **spatPomp**. In the remainder of this section, we describe and discuss some of the filtering methods currently implemented in the package.

3.3.1 The guided intermediate resampling filter (GIRF)

The guided intermediate resampling filter (GIRF, Park and Ionides, 2020) is an extension of the auxiliary particle filter (APF, Pitt and Shepard, 1999). GIRF is appropriate for moderately high-dimensional SpatPOMP models with a continuous-time latent process. All particle filters compute importance weights for proposed particles and carry out resampling to focus computational effort on particles consistent with the data (see reviews by Arulampalam et al., 2002; Doucet and Johansen, 2011; Kantas et al., 2015). In the context of **pomp**, the `pfilter` function is discussed by King et al. (2016). GIRF combines two techniques for improved scaling of particle filters: the use of a guide function and intermediate resampling.

The guide function steers particles using importance weights that anticipate upcoming observations. Future measurements are considered up to a lookahead horizon, L. APF corresponds to a lookahead horizon $L = 2$, and a basic particle filter has $L = 1$. Values $L \leq 3$ are typical for GIRF.

Intermediate resampling breaks each observation interval into S sub-intervals, and carries out reweighting and resampling on each sub-interval. Perhaps surprisingly, intermediate resampling can facilitate some otherwise intractable importance sampling problems (Del Moral and Murray, 2015). APF and the basic particle filter correspond to $S = 1$, whereas choosing $S = U$ gives favorable scaling properties (Park and Ionides, 2020).

In Algorithm 1 the F, G and P superscripts indicate filtered, guide and proposal particles, respectively. The goal for the pseudocode in Algorithm 1, and subsequent algorithms in this paper, is a succinct description of the logic of the procedure rather than a complete recipe for efficient coding. The code in **spatPomp** takes advantage of memory overwriting and vectorization opportunities that are not represented in this pseudocode.

Algorithm 1: `girf()` in **spatPomp**, run as `girf(P, Np =` J`, Ninter =` S`, Nguide =` K`, Lookahead =` L`)`, using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `dunit_measure`, `rinit`, `skeleton`, `obs` and `coef`.

```
input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1};\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0;\theta)$; evaluator for
       $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \mid x_{u,n};\theta)$, and $\boldsymbol{\mu}(\boldsymbol{x},s,t;\theta)$; data, $\boldsymbol{y}^*_{1:N}$; parameter, $\theta$; number of
       particles, $J$; number of guide simulations, $K$; number of intermediate timesteps, $S$;
       number of lookahead lags, $L$.
1  initialize: simulate $\boldsymbol{X}^{F,j}_{0,0} \sim f_{\boldsymbol{X}_0}(\,\cdot\,;\theta)$ and set $g^{F,j}_{0,0} = 1$ for $j$ in $1\!:\!J$
2  for $n$ in $0\!:\!N-1$ do
3      sequence of guide forecast times, $\mathbb{L} = (n+1)\!:\!\min(n+L,N)$
4      guide simulations, $\boldsymbol{X}^{G,j,k}_{\mathbb{L}} \sim f_{\boldsymbol{X}_{\mathbb{L}}|\boldsymbol{X}_n}\big(\,\cdot\,|\boldsymbol{X}^{F,j}_{n,0};\theta\big)$ for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$
5      guide residuals, $\boldsymbol{\epsilon}^{j,k}_{0,\ell} = \boldsymbol{X}^{G,j,k}_{\ell} - \boldsymbol{\mu}\big(\boldsymbol{X}^{F,j}_{n}, t_n, t_\ell;\theta\big)$ for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$, $\ell$ in $\mathbb{L}$
6      for $s$ in $1\!:\!S$ do
7          prediction simulations, $\boldsymbol{X}^{P,j}_{n,s} \sim f_{\boldsymbol{X}_{n,s}|\boldsymbol{X}_{n,s-1}}\big(\,\cdot\,|\boldsymbol{X}^{F,j}_{n,s-1};\theta\big)$ for $j$ in $1\!:\!J$
8          deterministic trajectory, $\boldsymbol{\mu}^{P,j}_{n,s,\ell} = \boldsymbol{\mu}\big(\boldsymbol{X}^{P,j}_{n,s}, t_{n,s}, t_\ell;\theta\big)$ for $j$ in $1\!:\!J$, $\ell$ in $\mathbb{L}$
9          pseudo guide simulations, $\hat{\boldsymbol{X}}^{j,k}_{n,s,\ell} = \boldsymbol{\mu}^{P,j}_{n,s,\ell} + \boldsymbol{\epsilon}^{j,k}_{s-1,\ell} - \boldsymbol{\epsilon}^{j,k}_{s-1,n+1} + \sqrt{\frac{t_{n+1}-t_{n,s}}{t_{n+1}-t_{n,0}}}\,\boldsymbol{\epsilon}^{j,k}_{s-1,n+1}$
            for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$, $\ell$ in $\mathbb{L}$
10         discount factor, $\eta_{n,s,\ell} = 1 - (t_{n+\ell} - t_{n,s})/\{(t_{n+\ell} - t_{\max(n+\ell-L,0)})\cdot(1+\mathbb{1}_{L=1})\}$
11         $g^{P,j}_{n,s} = \prod_{\ell\,\mathrm{in}\,\mathbb{L}} \prod_{u=1}^{U} \left[\frac{1}{K}\sum_{k=1}^{K} f_{Y_{u,\ell}|X_{u,\ell}}\Big(y^*_{u,\ell} \,|\, \hat{X}^{j,k}_{u,n,s,\ell};\theta\Big)\right]^{\eta_{n,s,\ell}}$ for $j$ in $1\!:\!J$
12         for $j$ in $1\!:\!J$, $w^j_{n,s} = \begin{cases} f_{\boldsymbol{Y}_n|\boldsymbol{X}_n}\big(\boldsymbol{y}_n \mid \boldsymbol{X}^{F,j}_{n,s-1};\theta\big)\; g^{P,j}_{n,s}\big/ g^{F,j}_{n,s-1} & \text{if } s=1 \text{ and } n\neq 0 \\ g^{P,j}_{n,s}\big/ g^{F,j}_{n,s-1} & \text{else} \end{cases}$
13         log likelihood component, $c_{n,s} = \log\Big(J^{-1}\sum_{k=1}^{J} w^k_{n,s}\Big)$
14         normalized weights, $\tilde{w}^j_{n,s} = w^j_{n,s}\Big/\sum_{k=1}^{J} w^k_{n,s}$ for $j$ in $1\!:\!J$
15         select resample indices, $r_{1:J}$ with $\mathbb{P}\,[r_j = k] = \tilde{w}^k_{n,s}$ for $j$ in $1\!:\!J$
16         $\boldsymbol{X}^{F,j}_{n,s} = \boldsymbol{X}^{P,r_j}_{n,s}$ , $g^{F,j}_{n,s} = g^{P,r_j}_{n,s}$ , $\boldsymbol{\epsilon}^{j,k}_{s,\ell} = \boldsymbol{\epsilon}^{r_j,k}_{s-1,\ell}$ for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$, $\ell$ in $\mathbb{L}$
17     end
18     set $\boldsymbol{X}^{F,j}_{n+1,0} = \boldsymbol{X}^{F,j}_{n,S}$ and $g^{F}_{n+1,0,j} = g^{F}_{n,S,j}$ for $j$ in $1\!:\!J$
19 end
output: log likelihood, $\ell^{\mathrm{GIRF}} = \sum_{n=0}^{N-1}\sum_{s=1}^{S} c_{n,s}$, and filter particles, $\boldsymbol{X}^{F,1:J}_{N,0}$
complexity: $\mathcal{O}\big(JLUN(K+S)\big)$
```

We call the guide in Algorithm 1 a *bootstrap guide function* since it is based on resampling the Monte Carlo residuals calculated in step 5. Another option of a guide function in `girf` is the simulated moment guide function developed by Park and Ionides (2020) which uses the `eunit_measure`, `vunit_measure` and `munit_measure` model components together with simulations to calculate the guide. The expectation of Monte Carlo likelihood estimates does not depend on the guide function, so an inexact guide approximation may lead to loss of numerical efficiency but does not affect the consistency of the procedure.

The intermediate resampling is represented in Algorithm 1 by the loop of $s = 1, \ldots, S$ in step 6. The intermediate times are defined by $t_{n,s} = t_n + (t_{n+1} - t_n) \cdot s/S$ and we write $\boldsymbol{X}_{n,s} = \boldsymbol{X}(t_{n,s})$. The resampling weights (step 12) are defined in terms of guide function evaluations $g^{P,j}_{n,s}$. The only requirement for the guide function to achieve unbiased estimates is that it satisfies $g^{F,j}_{0,0} = 1$ and $g^{P,j}_{N-1,S} = f_{\boldsymbol{Y}_N|\boldsymbol{X}_N}(\boldsymbol{y}^*_N \mid \boldsymbol{X}^{F,j}_{N-1,S}\,;\theta)$, which is the case in Algorithm 1. The particular guide function calculated in step 11 evaluates particles using a prediction centered on a function

$$\boldsymbol{\mu}(\boldsymbol{x}, s, t\,;\theta) \approx \mathbb{E}[\boldsymbol{X}(t) \mid \boldsymbol{X}(s) = \boldsymbol{x}\,;\theta].$$

We call $\boldsymbol{\mu}(\boldsymbol{x}, s, t\,;\theta)$ a *deterministic trajectory* associated with $\boldsymbol{X}(t)$. For a continuous time SpatPOMP model, this trajectory is typically the solution to a system of differential equations that define a vector field called the *skeleton* (Tong, 1990). In **spatPomp**, the argument to `skeleton` is a map or vector field which is numerically solved to obtain $\boldsymbol{\mu}(\boldsymbol{x}, s, t\,;\theta)$. It can be specified using complied `C` code via a `C` snippet argument to `spatPomp`, as demonstrated in Section 3.6. The forecast spread around this point prediction is given by the simulated bootstrap residuals constructed in step 5.

3.3.2 Adapted bagged filter (ABF)

The adapted bagged filter (Ionides et al., 2021) combines many independent particle filters. This is called *bagging*, (*b*ootstrap *agg*regat*ing*), since a basic particle filter is also called a bootstrap filter. The adapted distribution is the conditional distribution of the latent process given its current value and the subsequent observation (Johansen and Doucet, 2008). In the adapted bagged filter, each bootstrap replicate makes a Monte Carlo approximation to a draw from the adapted distribution. Thus, in the pseudocode of Algorithm 2, $\boldsymbol{X}^{A,i}_{0:N}$ is a Monte Carlo sample targeting the adapted sampling distribution,

$$f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)\prod_{n=1}^{N} f_{\boldsymbol{X}_n|\boldsymbol{Y}_n,\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n\,|\,\boldsymbol{y}^*_n,\boldsymbol{x}_{n-1}\,;\theta). \tag{3.1}$$

Each adapted simulation replicate is constructed by importance sampling using proposal particles $\{\boldsymbol{X}^{P,i,j}_n\}$. The ensemble of adapted simulation replicates are then weighted using data in a spatiotemporal neighborhood of each observation to obtain a locally combined Monte Carlo sample targeting the filter distribution, with some approximation error due to the finite spatiotemporal neighborhood used. This local aggregation of the bootstrap replicates also provides an evaluation of the likelihood function.

On a given bootstrap replicate i at a given time n, all the adapted proposal particles $\boldsymbol{X}^{P,i,1:J}_n$ in step 3 are necessarily close to each other in state space because they share the parent particle $\boldsymbol{X}^{A,i}_{n-1}$. This reduces imbalance in the adapted weights in step 5, which helps to battle the curse of dimensionality that afflicts importance sampling. The combination of the replicates for the filter estimate in step 11 is carried out using only weights in a spatiotemporal neighborhood, thus avoiding the curse of dimensionality. For any point (u,n), the neighborhood $B_{u,n}$ should be specified as a subset of $A_{u,n} = \{(\tilde{u}, \tilde{n}) : \tilde{n} < n \text{ or } (\tilde{u} < u \text{ and } \tilde{n} = n)\}$. If the model has a mixing property, meaning that conditioning on the observations in the neighborhood $B_{u,n}$ is negligibly different from conditioning on the full set $A_{u,n}$, then the approximation involved in this localization is adequate.

Steps 1 through 7 do not involve interaction between replicates and therefore iteration over i can be carried out in parallel. If a parallel backend has been set up by the user, the `abf` method will parallelize computations over the replicates using multiple cores. The user can register a parallel backend using the `doParallel` package (Wallig and Weston, 2020, 2019) prior to calling `abf`.

```
R> library("doParallel")
R> registerDoParallel(3) # Parallelize over 3 cores
```

The neighborhood is supplied via the `nbhd` argument to `abf` as a function which takes a point in space-time, (u, n), and returns a list of points in space-time which correspond to $B_{u,n}$. An example with $B_{u,n} = \{(u-1, n), (u, n-1)\}$ follows.

```
R> example_nbhd <- function(object, unit, time){
+    nbhd_list = list()
+    if(time>1) nbhd_list <- c(nbhd_list, list(c(unit, time-1)))
+    if(unit>1) nbhd_list <- c(nbhd_list, list(c(unit-1, time)))
+    return(nbhd_list)
+ }
```

ABF can be combined with the guided intermediate resampling technique used by GIRF to give an algorithm called ABF-IR (Ionides et al., 2021) implemented as `abfir`.

Algorithm 2: `abf()` in **spatPomp**, run as `abf(P, replicates =`$\mathcal{I}$`, Np =`J`, nbhd=`$B_{u,n}$`)`, using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `dunit_measure`, `rinit`, `obs` and `coef`.

```
   input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}\,;\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$; evaluator for
          $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \mid x_{u,n}\,;\theta)$; data, $\boldsymbol{y}^*_{1:N}$; parameter, $\theta$; number of particles per replicate,
          $J$; number of replicates, $\mathcal{I}$; neighborhood structure, $B_{u,n}$
1  initialize adapted simulation, $\boldsymbol{X}_0^{A,i} \sim f_{\boldsymbol{X}_0}(\cdot\,;\theta)$ for $i$ in $1\!:\!\mathcal{I}$
2  for $n$ in $1\!:\!N$ do
3      proposals, $\boldsymbol{X}_n^{P,i,j} \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}\big(\cdot \mid \boldsymbol{X}_{n-1}^{A,i}\,;\theta\big)$ for $i$ in $1\!:\!\mathcal{I}$, $j$ in $1\!:\!J$
4      $w_{u,n}^{i,j} = f_{Y_{u,n}|X_{u,n}}\big(y^*_{u,n} \mid X_{u,n}^{P,i,j}\,;\theta\big)$ for $u$ in $1\!:\!U$, $i$ in $1\!:\!\mathcal{I}$, $j$ in $1\!:\!J$
5      adapted resampling weights, $w_n^{A,i,j} = \prod_{u=1}^{U} w_{u,n}^{i,j}$ for $u$ in $1\!:\!U$, $i$ in $1\!:\!\mathcal{I}$, $j$ in $1\!:\!J$
6      set $\boldsymbol{X}_n^{A,i} = \boldsymbol{X}_n^{P,i,j}$ with probability $w_n^{A,i,j}\left(\sum_{k=1}^{J} w_n^{A,i,k}\right)^{-1}$ for $i$ in $1\!:\!\mathcal{I}$
7      $w_{u,n}^{P,i,j} = \prod_{\tilde{n}=1}^{n-1}\left[\frac{1}{J}\sum_{k=1}^{J}\prod_{(\tilde{u},\tilde{n})\in B_{u,n}} w_{\tilde{u},\tilde{n}}^{i,k}\right]\prod_{(\tilde{u},n)\in B_{u,n}} w_{\tilde{u},n}^{i,j}$ for $u$ in $1\!:\!U$, $i$ in $1\!:\!\mathcal{I}$, $j$ in $1\!:\!J$
8  end
9  filter weights, $w_{u,n}^{F,i,j} = \dfrac{w_{u,n}^{i,j}\; w_{u,n}^{P,i,j}}{\sum_{p=1}^{\mathcal{I}}\sum_{k=1}^{J} w_{u,n}^{P,p,k}}$ for $u$ in $1\!:\!U$, $n$ in $1\!:\!N$, $i$ in $1\!:\!\mathcal{I}$, $j$ in $1\!:\!J$
10 conditional log likelihood, $\ell_{u,n} = \log\left(\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w_{u,n}^{F,i,j}\right)$ for $u$ in $1\!:\!U$, $n$ in $1\!:\!N$
11 set $X_{u,n}^{F,j} = X_{u,n}^{P,i,k}$ with probability $w_{u,n}^{F,i,k}\, e^{-\ell_{u,n}}$ for $u$ in $1\!:\!U$, $n$ in $1\!:\!N$, $j$ in $1\!:\!J$
   output: filter particles, $\boldsymbol{X}_n^{F,1:J}$, for $n$ in $1\!:\!N$; log likelihood, $\ell^{\text{ABF}} = \sum_{n=1}^{N}\sum_{u=1}^{U}\ell_{u,n}$
   complexity: $\mathcal{O}(\mathcal{I}JUN)$
```

Algorithm 3: `enkf()` in **spatPomp**, run as `enkf(P, Np =` J`)`, using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `eunit_measure`, `vunit_measure`, `rinit`, `coef` and `obs`.

```
   input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{x}_{n-1}\,;\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$; evaluator for $\mathrm{e}_u(X_{u,n},\theta)$ and
          $\mathrm{v}_u(X_{u,n},\theta)$; parameter, $\theta$; data, $\boldsymbol{y}^*_{1:N}$; number of particles, $J$.
1  initialize filter particles, $\boldsymbol{X}^{F,j}_0 \sim f_{\boldsymbol{X}_0}(\,\cdot\,;\theta)$ for $j$ in $1\,{:}\,J$
2  for $n$ in $1\,{:}\,N$ do
3      prediction ensemble, $\boldsymbol{X}^{P,j}_n \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}\big(\,\cdot\,|\boldsymbol{X}^{F,j}_{n-1};\theta\big)$ for $j$ in $1\,{:}\,J$
4      centered prediction ensemble, $\tilde{\boldsymbol{X}}^{P,j}_n = \boldsymbol{X}^{P,j}_n - \frac{1}{J}\sum_{k=1}^{J}\boldsymbol{X}^{P,k}_n$ for $j$ in $1\,{:}\,J$
5      forecast ensemble, $\hat{\boldsymbol{Y}}^j_n = \mathrm{e}_u(X^{P,j}_{u,n},\theta)$ for $j$ in $1\,{:}\,J$
6      forecast mean, $\overline{\boldsymbol{Y}}_n = \frac{1}{J}\sum_{j=1}^{J}\hat{\boldsymbol{Y}}^j_n$
7      centered forecast ensemble, $\tilde{\boldsymbol{Y}}^j_n = \hat{\boldsymbol{Y}}^j_n - \overline{\boldsymbol{Y}}_n$ for $j$ in $1\,{:}\,J$
8      forecast measurement variance, $R_{u,\tilde{u}} = \mathbb{1}_{u,\tilde{u}}\,\frac{1}{J}\sum_{j=1}^{J}\mathrm{v}_u\big(\boldsymbol{X}^{P,j}_{u,n},\theta\big)$ for $u,\tilde{u}$ in $1\,{:}\,U$
9      forecast estimated covariance, $\Sigma_Y = \frac{1}{J-1}\sum_{j=1}^{J}(\tilde{\boldsymbol{Y}}^j_n)(\tilde{\boldsymbol{Y}}^j_n)^T + R$
10     prediction and forecast sample covariance, $\Sigma_{XY} = \frac{1}{J-1}\sum_{j=1}^{J}(\tilde{\boldsymbol{X}}^{P,j}_n)(\tilde{\boldsymbol{Y}}^j_n)^T$
11     Kalman gain, $K = \Sigma_{XY}\Sigma_Y^{-1}$
12     artificial measurement noise, $\boldsymbol{\epsilon}^j_n \sim \mathrm{Normal}(\boldsymbol{0},R)$ for $j$ in $1\,{:}\,J$
13     errors, $\boldsymbol{r}^j_n = \hat{\boldsymbol{Y}}^j_n - \boldsymbol{y}^*_n$ for $j$ in $1\,{:}\,J$
14     filter update, $\boldsymbol{X}^{F,j}_n = \boldsymbol{X}^{P,j}_n + K\big(\boldsymbol{r}^j_n + \boldsymbol{\epsilon}^j_n\big)$ for $j$ in $1\,{:}\,J$
15     $\ell_n = \log\big[\phi\big(\boldsymbol{y}^*_n\,;\overline{\boldsymbol{Y}}_n,\Sigma_Y\big)\big]$ where $\phi(\cdot\,;\boldsymbol{\mu},\Sigma)$ is the Normal$(\boldsymbol{\mu},\Sigma)$ density.
16 end
   output: filter sample, $\boldsymbol{X}^{F,1:J}_n$, for $n$ in $1\,{:}\,N$; log likelihood estimate, $\ell^{\mathrm{EnKF}} = \sum_{n=1}^{N}\ell_n$
   complexity: $\mathcal{O}(JUN)$
```

3.3.3 The ensemble Kalman filter (EnKF)

Algorithm 3 is an implementation of the ensemble Kalman filter (EnKF)(Evensen, 1994; Evensen and van Leeuwen, 1996). The EnKF makes a Gaussian approximation to assimilate Monte Carlo simulations from a state prediction model with data observed in the corresponding time step. The EnKF has two steps: the prediction and the filtering steps; the latter is known the "analysis" step in the geophysical data-assimilation literature. The prediction step advances Monte Carlo particles to the next observation time step by using simulations from a possibly non-linear transition model. In the filtering step, the sample estimate of the state covariance matrix and the linear measurement model are used to make a Gaussian approximation to the conditional distribution of the state given the data.

In step 8 of Algorithm 3, the conditional variance of the measurement at the current time step is approximated by constructing a diagonal covariance matrix whose diagonal elements are the sample average of the theoretical unit measurement variances at each unit. This is written using an indicator function $\mathbb{1}_{u,\tilde{u}}$ which takes value 1 if $u = \tilde{u}$ and 0 otherwise. The `vunit_measure` model component aids in this step whereas `eunit_measure` specifies how we can construct forecast data (step 5) that can be used to later update our prediction particles in step 14. In step 12 we add artificial measurement error to arrive at a consistent sample covariance for the filtering step (Evensen, 1994; Evensen and van Leeuwen, 1996), writing Normal$(\boldsymbol{\mu}, \Sigma)$ for independent draws from a multivariate normal random variable with mean $\boldsymbol{\mu}$ and variance matrix Σ.

EnKF achieves good dimensional scaling relative to a particle filter (by avoiding the resampling step) at the expense of a Gaussian approximation in the filtering update rule. Adding hierarchical layers to the model representation can help to make the EnKF approximation applicable in non-

Gaussian contexts (Katzfuss et al., 2020). Since we envisage **spatPomp** primarily for situations with relatively low dimension, this implementation does not engage in regularization issues required when the dimension of the observation space exceeds the number of particles in the ensemble.

Our EnKF implementation supposes we have access to the functions

$$\begin{aligned} \mathrm{e}_{u,n}(x,\theta) &= \mathbb{E}\big[Y_{u,n} \,|\, X_{u,n}{=}x\,;\theta\big] \\ \mathrm{v}_{u}(x,\theta) &= \mathrm{Var}\big(Y_{u,n} \,|\, X_{u,n}{=}x\,;\theta\big) \end{aligned}$$

If the measurements are unbiased, $\mathrm{e}_{u,n}(x_u,\theta)$ will simply extract the measured components of x_u. The measurements in a SpatPOMP do not necessarily correspond to specific components of the state vector, and `enkf` permits arbitrary relationships subject to the constraint that the user can provide the necessary $\mathrm{e}_{u,n}(x,\theta)$ and $\mathrm{v}_u(x,\theta)$ functions. These functions can be defined during the construction of a class '`spatPomp`' object by supplying `C` snippets to the arguments `eunit_measure` and `vunit_measure` respectively. For common choices of measurement model, such as Gaussian or negative binomial, $\mathrm{e}_{u,n}$ and $\mathrm{v}_{u,n}$ are readily available. In general, the functional forms of $\mathrm{e}_{u,n}$ and $\mathrm{v}_{u,n}$ may depend on u and n. Users can specify more general functional forms in **spatPomp** since the variables `u` and `t` are defined for the C snippets. Similarly, both the mathematical notation in Section 3.2 and the **spatPomp** implementation permit arbitrary dependence on covariate time series.

3.3.4 Block particle filter

Algorithm 4 is an implementation of the block particle filter (BPF Rebeschini and van Handel, 2015), also called the factored particle filter (Ng et al., 2002). BPF partitions the units into a collection of blocks, $\mathcal{B}_1, \ldots, \mathcal{B}_K$. Each unit is placed in exactly one block. BPF generates proposal particles by simulating from the joint latent process across all blocks, exactly as the particle filter does. However, the resampling in the filtering step is carried out independently for each block, using weights corresponding only to the measurements in the block. Different proposal particles may be successful for different blocks, and the block resampling allows the filter particles to paste together these successful proposed blocks. BPF supposes that spatial coupling conditional on the data occurs primarily within blocks, and is negligible between blocks.

The user has a choice of specifying the blocks using either the `block_list` argument or `block_size`, but not both. `block_list` takes a class '`list`' object where each entry is a vector representing the units in a block. `block_size` takes an integer and evenly partitions $1\!:\!U$ into blocks of size approximately `block_size`. For example, if there are 4 units, executing `bpfilter` with `block_size=2` is equivalent to setting `block_list=list(c(1,2),c(3,4))`.

3.4 Inference for SpatPOMP models

We focus on iterated filtering methods (Ionides et al., 2015) which provide a relatively simple way to coerce filtering algorithms to carry out parameter inference, applicable to the general class of SpatPOMP models considered by **spatPomp**. The main idea of iterated filtering is to extend a POMP model to include dynamic parameter perturbations. Repeated filtering, with parameter perturbations of decreasing magnitude, approaches the maximum likelihood estimate. Here, we present iterated versions of GIRF, EnKF and the unadapted bagged filter (UBF), a version of ABF with $J = 1$. These algorithms are known as IGIRF (Park and Ionides, 2020), IEnKF (Li et al., 2020) and IUBF (Ionides et al., 2021) respectively. SpatPOMP model estimation is an active area for research (for example, Katzfuss et al., 2020) and **spatPomp** provides a platform for developing and testing new methods, in addition to new models and data analysis.

3.4.1 Iterated GIRF for parameter estimation

Algorithm 5 describes `igirf()`, the **spatPomp** implementation of IGIRF. This algorithm carries out the IF2 algorithm of Ionides et al. (2015) with filtering carried out by GIRF, therefore its implementation combines the `mif2` function in **pomp** with `girf` (Algorithm 1). For Algorithm 5, we unclutter the pseudocode by using a subscript and superscript notation for free indices, meaning subscripts and superscripts for which a value is not explicitly specified in the code. We use the convention that a free subscript or superscript is evaluated for all values in its range, leading to an implicit 'for' loop. This does not hold for capitalized subscripts and superscripts, which describe the purpose of a Monte Carlo particle, matching usage in Algorithm 1.

The quantity $\Theta^{P,m,j}_{n,s}$ gives a perturbed parameter vector for θ corresponding to particle j on iteration m at the s^{th} intermediate time between n and $n+1$. The perturbations in Algorithm 5 are taken to follow a multivariate normal distribution, with a diagonal covariance matrix scaled by σ_{n,d_θ}. Normal perturbations are not theoretically required, but this is a common choice in practice. The `igirf` function permits perturbations to be carried out on a transformed scale, specified using the `partrans` argument, to accommodate situations where normally distributed perturbations are more natural on the log or logistic scale, or any other user-specified scale. For regular parameters, i.e. parameters that are not related to the initial conditions of the dynamics, it may be appropriate to set the perturbation scale independent of n. If parameters are transformed so that a unit scale is relevant, for example using a logarithmic transform for non-negative parameters, a simple choice such as $\sigma_{n,d_\theta} = 0.02$ may be effective. Initial value parameters (IVPs) are those that determine only the latent state at time t_0, and these should be perturbed only at the beginning of each iteration m. The matrix $\sigma_{0:N,1:D_\theta}$ can be constructed using the `rw.sd` function, which simplifies the construction for regular parameters and IVPs. The `cooling.fraction.50` argument takes the fraction of `rw.sd` by which to perturb the parameters after 50 iterations of `igirf`. If set to 0.5, for instance, the default behavior is to lower the perturbation standard deviation geometrically so that it is halved by the 50th iteration of `igirf`.

3.4.2 Iterated EnKF for parameter estimation

Algorithm 6 is an implementation of the iterated ensemble Kalman filter (IEnKF) which extends the IF2 approach for parameter estimation by replacing a particle filter with an ensemble Kalman filter. As described in Section 3.4.1, we employ a free index notation whereby superscripts and subscripts that are not otherwise specified have an implicit 'for' loop.

We note a caveat in using IEnKF. If the forecast mean $\hat{\boldsymbol{Y}}$ is not dependent on a parameter component, that component of the parameter is not updated by the Kalman gain on average. For example, in Brownian motion, the forecast $\hat{\boldsymbol{Y}}$ is independent of the measurement variance parameter τ, and so IEnKF is ineffective in estimating τ. By contrast, for geometric Brownian motion, which is obtained by exponentiating Brownian motion, IEnKF can estimate τ because high values of τ lead to higher values of $\hat{\boldsymbol{Y}}$ on average. In this case, if the average forecast is different from the observed data, the τ parameter gets updated accordingly to reduce the error. Therefore, IEnKF may need to be coupled with other parameter estimation methods (such as IGIRF) to estimate parameters that do not affect the forecast mean.

3.4.3 Iterated UBF for parameter estimation

Algorithm 7 also extends the IF2 approach by using an ABF-inspired particle filter. We start with N_θ copies of our starting parameter set and iteratively perturb the parameter set and evaluate a conditional likelihood at each observation time using ABF with $J = 1$ (also called the unadapted bagged filter, or UBF). The parameter sets yielding the top p quantile of the likelihoods are resampled for pertubation and likelihood evaluation in the next time step.

3.4.4 Inference algorithms inherited from pomp

Objects of class '`spatPomp`' inherit methods for inference from class '`pomp`' objects implemented in the **pomp** package. As discussed earlier, IF2 (Ionides et al., 2015) enables parameter estimation in the frequentist sense and has been used in numerous applications. It can be used to check the capabilities of newer and more scalable inference methods on smaller examples for which IF2 is known to be effective. Extensions for Bayesian inference of the currently implemented high-dimensional particle filter methods (GIRF, ABF, EnKF, BPF) are not yet available. Bayesian inference is available in **spatPomp** using the approximate Bayesian computing (ABC) method inherited from **pomp**, `abc()`. ABC has previously been used for spatiotemporal inference (Brown et al., 2018) and can also serve as a baseline method. However, ABC is a feature-based method that may lose substantial information compared to full-information methods that work with the full likelihood function.

3.5 Demonstrating data analysis tools on a toy model

We illustrate key capabilities of **spatPomp** using a model for correlated Brownian motions. This allows us to demonstrate a data analysis in a simple context where we can compare results with a standard particle filter as well as validate all methods against the exact solutions which are analytically available. Here we defer the details of model construction by using a model prespecified within the package. Section 3.6 proceeds to develop a model exemplifying the kinds of nonlinear, non-Gaussian spatiotemporal dynamics of moderately high dimension, which are the target of **spatPomp**. Consider spatial units $1, \dots, U$ located evenly around a circle, with $\text{dist}(u, \tilde{u})$ being the circle distance,

$$\text{dist}(u, \tilde{u}) = \min\big(|u - \tilde{u}|, |u - \tilde{u} + U|, |u - \tilde{u} - U|\big).$$

We investigate a SpatPOMP where the latent process is a U-dimensional Brownian motion $\boldsymbol{X}(t)$ having correlation that decays with distance. Specifically,

$$dX_u(t) = \sum_{\tilde{u}=1}^{U} \rho^{\text{dist}(u,\tilde{u})} dW_{\tilde{u}}(t),$$

where $W_1(t), \dots, W_U(t)$ are independent Brownian motions with infinitesimal variance σ^2, and $|\rho| < 1$. Using the notation in Section 3.2, we suppose our measurement model for discrete-time observations of the latent process is

$$Y_{u,n} = X_{u,n} + \eta_{u,n}$$

where $\eta_{u,n} \overset{\text{iid}}{\sim} \text{Normal}(0, \tau^2)$. The model specification is completed by specifying the initial conditions, $\{X_u(0), u \in 1 : U\}$. A class '`spatPomp`' object which simulates from this model for $U = 10$ with $N = 20$ evenly-spaced observations that are one unit time apart can be simulated using `bm()` and plotted using `plot()`, yielding the plot in Figure 3.3 as follows:

Algorithm 4: `bpfilter()` in **spatPomp**, run as `bpfilter(P, Np =` J`, block_list =` $\mathcal{B}$`)` using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `dunit_measure`, `rinit`, `obs`, `coef`.

```
  input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{x}_{n-1}\,;\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$; number of particles, $J$;
         evaluator for $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \,|\, x_{u,n}\,;\theta)$; data, $\boldsymbol{y}^*_{1:N}$; parameter, $\theta$; blocks, $\mathcal{B}_{1:K}$;
1 initialization, $\boldsymbol{X}_0^{F,j} \sim f_{\boldsymbol{X}_0}(\,\cdot\,;\theta)$ for $j$ in $1\!:\!J$
2 for $n$ in $1\!:\!N$ do
3     prediction, $\boldsymbol{X}_n^{P,j} \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}\big(\,\cdot\,|\boldsymbol{X}_{n-1}^{F,j};\theta\big)$ for $j$ in $1\!:\!J$
4     block weights, $w_{k,n}^j = \prod_{u\in\mathcal{B}_k} f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, X_{u,n}^{P,j}\,;\theta)$ for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$
5     resampling indices, $r_{k,n}^j$ with $\mathbb{P}\left[r_{k,n}^j = i\right] = \tilde{w}_{k,n}^i \Big/ \sum_{q=1}^J w_{k,n}^q$ for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$
6     resample, $X_{\mathcal{B}_k,n}^{F,j} = X_{\mathcal{B}_k,n}^{P,r_{j,k}}$ for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$
7 end
  output: log likelihood, $\ell^{\text{BPF}}(\theta) = \sum_{n=1}^N \sum_{k=1}^K \log\Big(\frac{1}{J}\sum_{j=1}^J w_{k,n}^j\Big)$, filter particles $\boldsymbol{X}_{1:N}^{F,1:J}$
  complexity: $\mathcal{O}(JUN)$
```

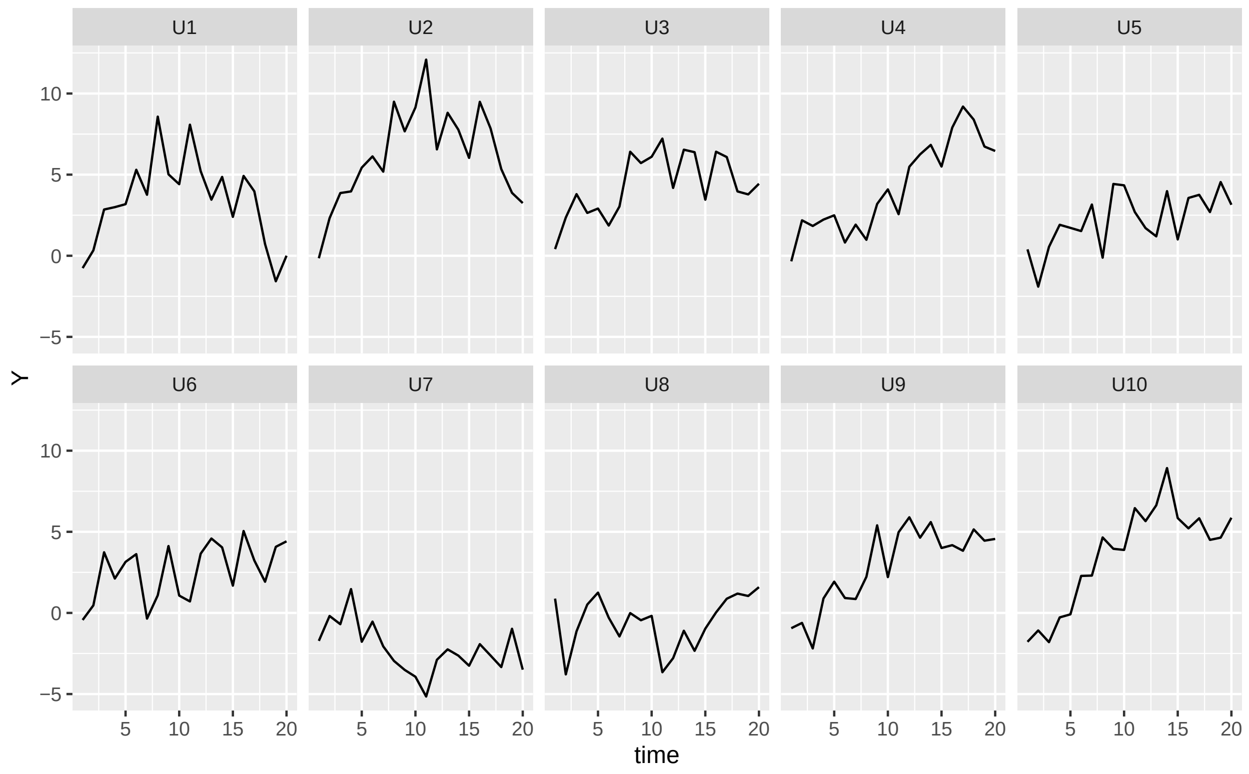


Figure 3.3 – Output of the `plot()` method on a class '`spatPomp`' object. This object represents a simulation from a 10-dimensional correlated Brownian motions model with 20 observations that are one unit time apart.

Algorithm 5: `igirf()` in **spatPomp**, run as `igirf(P, params =`θ_0`, Ngirf =`M`, Np =`J`, Ninter =`S`, Nguide =`K`, Lookahead =`L`), rw.sd =`$\sigma_{0:N,1:D_\theta}$`, cooling.fraction.50 =`a using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `dunit_measure`, `skeleton`, `rinit` and `obs`

```
input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}\,;\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$; evaluator for
       $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \mid x_{u,n}\,;\theta)$, and $\boldsymbol{\mu}(\boldsymbol{x}, s, t\,;\theta)$; data, $\boldsymbol{y}^*_{1:N}$; starting parameter, $\theta_0$;
       iterations, $M$; particles, $J$; lookahead lags, $L$; intermediate timesteps, $S$; random
       walk intensities, $\sigma_{0:N,1:D_\theta}$; cooling fraction in 50 iterations, $a$.
note: free indices are implicit 'for' loops, calculated for $j$ in $1\!:\!J$, $k$ in $1\!:\!K$,
      $\ell$ in $(n+1)\!:\!\min(n+L, N)$, $u$ in $1\!:\!U$, $d_\theta, d'_\theta$ in $1\!:\!D_\theta$.
1  initialize parameters, $\Theta^{F,0,j}_{N-1,S} = \theta_0$
2  for $m$ in $1\!:\!M$ do
3      initialize parameters, $\Theta^{F,m,j}_{0,0} \sim \mathrm{Normal}\big(\Theta^{F,m-1,j}_{N-1,S}\,, a^{2m/50}\,\Sigma_{\mathrm{ivp}}\big)$ for
         $\big[\Sigma_{\mathrm{ivp}}\big]_{d_\theta,d'_\theta} = \sigma^2_{\mathrm{ivp},d_\theta}\mathbb{1}_{d_\theta,d'_\theta}$
4      initialize filter particles, simulate $\boldsymbol{X}^{F,j}_{0,0} \sim f_{\boldsymbol{X}_0}\left(\,\cdot\,;\Theta^{F,m,j}_{0,0}\right)$ and set $g^{F,j}_{n,0} = 1$
5      for $n$ in $0\!:\!N-1$ do
6          guide simulations, $\boldsymbol{X}^{G,j,k}_{\ell} \sim f_{\boldsymbol{X}_\ell|\boldsymbol{X}_n}\big(\,\cdot\,|\boldsymbol{X}^{F,j}_{n,0}\,;\Theta^{F,m,j}_{n,0}\big)$
7          guide residuals, $\boldsymbol{\epsilon}^{j,k}_{0,\ell} = \boldsymbol{X}^{G,j,k}_{\ell} - \boldsymbol{\mu}\big(\boldsymbol{X}^{F,j}_{n,0}, t_n, t_\ell\,;\Theta^{F,m,j}_{n,0}\big)$
8          for $s$ in $1\!:\!S$ do
9              perturb parameters, $\Theta^{P,m,j}_{n,s} \sim \mathrm{Normal}\big(\Theta^{F,m,j}_{n,s-1}\,, a^{2m/50}\,\Sigma_n\big)$ for
                 $\big[\Sigma_n\big]_{d_\theta,d'_\theta} = \sigma^2_{n,d_\theta}\mathbb{1}_{d_\theta,d'_\theta}/S$
10             prediction simulations, $\boldsymbol{X}^{P,j}_{n,s} \sim f_{\boldsymbol{X}_{n,s}|\boldsymbol{X}_{n,s-1}}\big(\,\cdot\,|\boldsymbol{X}^{F,j}_{n,s-1}\,;\Theta^{P,m,j}_{n,s}\big)$
11             deterministic trajectory, $\boldsymbol{\mu}^{P,j}_{n,s,\ell} = \boldsymbol{\mu}\big(\boldsymbol{X}^{P,j}_{n,s}, t_{n,s}, t_\ell\,;\Theta^{P,m,j}_{n,s}\big)$
12             pseudo guide simulations,
                 $\hat{\boldsymbol{X}}^{j,k}_{n,s,\ell} = \boldsymbol{\mu}^{P,j}_{n,s,\ell} + \boldsymbol{\epsilon}^{j,k}_{s-1,\ell} - \boldsymbol{\epsilon}^{j,k}_{s-1,n+1} + \sqrt{\frac{t_{n+1}-t_{n,s}}{t_{n+1}-t_{n,0}}}\,\boldsymbol{\epsilon}^{j,k}_{s-1,n+1}$
13             discount factor, $\eta_{n,s,\ell} = 1 - (t_{n+\ell} - t_{n,s})/\{(t_{n+\ell} - t_{\max(n+\ell-L,0)})\cdot(1 + \mathbb{1}_{L=1})\}$
14             $g^{P,j}_{n,s} = \prod_{\ell=n+1}^{\min(n+L,N)} \prod_{u=1}^{U} \left[\frac{1}{K}\sum_{k=1}^{K} f_{Y_{u,\ell}|X_{u,\ell}}\Big(y^*_{u,\ell} \mid \hat{X}^{j,k}_{u,n,s,\ell}\,;\Theta^{P,m,j}_{n,s}\Big)\right]^{\eta_{n,s,\ell}}$
15             $w^j_{n,s} = \begin{cases} f_{\boldsymbol{Y}_n|\boldsymbol{X}_n}\big(\boldsymbol{y}_n \mid \boldsymbol{X}^{F,j}_{n,s-1}\,;\Theta^{F,m,j}_{n,s-1}\big)\; g^{P,j}_{n,s}\Big/ g^{F,j}_{n,s-1} & \text{if } s=1,\ n\neq 0 \\ g^{P,j}_{n,s}\Big/ g^{F,j}_{n,s-1} & \text{else} \end{cases}$
16             normalized weights, $\tilde{w}^j_{n,s} = w^j_{n,s}\Big/\sum_{k=1}^{J} w^k_{n,s}$
17             resampling indices, $r_{1:J}$ with $\mathbb{P}\left[r_j = k\right] = \tilde{w}^k_{n,s}$
18             set $\boldsymbol{X}^{F,j}_{n,s} = \boldsymbol{X}^{P,r_j}_{n,s}$ , $g^{F,j}_{n,s} = g^{P,r_j}_{n,s}$ , $\boldsymbol{\epsilon}^{j,k}_{s,\ell} = \boldsymbol{\epsilon}^{r_j,k}_{s-1,\ell}$, $\Theta^{F,m,j}_{n,s} = \Theta^{P,m,r_j}_{n,s}$
19         end
20     end
21 end
output: Iterated GIRF parameter swarm, $\Theta^{F,M,1:J}_{N-1,S}$
Monte Carlo maximum likelihood estimate: $\frac{1}{J}\sum_{j=1}^{J}\Theta^{F,M,j}_{N-1,S}$
complexity: $\mathcal{O}\big(MJLUN(K+S)\big)$
```

Algorithm 6: `ienkf()` in **spatPomp**, run as `ienkf(P, params =`θ_0`, Nenkf =`M`, Np =`J`, cooling.fraction.50 =`a`, rw.sd =`$\sigma_{0:N,1:D_\theta}$`)`, using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `eunit_measure`, `vunit_measure`, `rinit`, and `obs`.

```
input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}\,;\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$; evaluator for $\mathrm{e}_{u,n}(x,\theta)$ and
       $\mathrm{v}_{u,n}(x,\theta)$; data, $\boldsymbol{y}^*_{1:N}$; number of particles, $J$; number of iterations, $M$; starting
       parameter, $\theta_0$; random walk intensities, $\sigma_{0:N,1:D_\theta}$; cooling fraction in 50 iterations,
       $a$.
note: free indices are implicit 'for' loops, calculated for $j$ in $1\!:\!J$, $u$ and $\tilde{u}$ in $1\!:\!U$, $d_\theta$ and
      $d'_\theta$ in $1\!:\!D_\theta$.
1  initialize parameters, $\Theta^{F,0,j}_N = \theta_0$
2  for $m$ in $1\!:\!M$ do
3      initialize parameters, $\Theta^{F,m,j}_0 \sim \mathrm{Normal}\big(\Theta^{F,m-1,j}_N\ , a^{2m/50}\,\Sigma_0\big)$ for $\big[\Sigma_n\big]_{d_\theta,d'_\theta} = \sigma^2_{n,d_\theta}\mathbb{1}_{d_\theta,d'_\theta}$
4      initialize filter particles, simulate $\boldsymbol{X}^{F,j}_0 \sim f_{\boldsymbol{X}_0}\Big(\,\cdot\,;\Theta^{F,m,j}_0\Big)$.
5      for $n$ in $1\!:\!N$ do
6          perturb parameters, $\Theta^{P,m,j}_n \sim \mathrm{Normal}\big(\Theta^{F,m,j}_{n-1}\ , a^{2m/50}\,\Sigma_n\big)$
7          prediction ensemble, $\boldsymbol{X}^{P,j}_n \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}\big(\,\cdot\,|\boldsymbol{X}^{F,j}_{n-1};\Theta^{P,m,j}_n\big)$
8          process and parameter ensemble, $\boldsymbol{Z}^{P,j}_n = \begin{pmatrix}\boldsymbol{X}^{P,j}_n \\ \Theta^{P,m,j}_n\end{pmatrix}$
9          centered process and parameter ensemble, $\tilde{\boldsymbol{Z}}^{P,j}_n = \boldsymbol{Z}^{P,j}_n - \frac{1}{J}\sum_{k=1}^{J}\boldsymbol{Z}^{P,k}_n$
10         forecast ensemble, $\hat{Y}^j_{u,n} = \mathrm{e}_u(X^{P,j}_{u,n},\Theta^{P,m,j}_n)$
11         centered forecast ensemble, $\tilde{\boldsymbol{Y}}^j_n = \hat{\boldsymbol{Y}}^j_n - \frac{1}{J}\sum_{k=1}^{J}\hat{\boldsymbol{Y}}^k_n$
12         forecast measurement variance, $R_{u,\tilde{u}} = \mathbb{1}_{u,\tilde{u}}\,\frac{1}{J}\sum_{j=1}^{J}\mathrm{v}_u(\boldsymbol{X}^{P,j}_{u,n},\Theta^{P,m,j}_n)$
13         forecast sample covariance, $\Sigma_Y = \frac{1}{J-1}\sum_{j=1}^{J}(\tilde{\boldsymbol{Y}}^j_n)(\tilde{\boldsymbol{Y}}^j_n)^T + R$
14         prediction and forecast sample covariance, $\Sigma_{ZY} = \frac{1}{J-1}\sum_{j=1}^{J}(\tilde{\boldsymbol{Z}}^{P,j}_n)(\tilde{\boldsymbol{Y}}^j_n)^T$
15         Kalman gain: $K = \Sigma_{ZY}\Sigma_Y^{-1}$
16         artificial measurement noise, $\boldsymbol{\epsilon}^j_n \sim \mathrm{Normal}(\boldsymbol{0},R)$
17         errors, $\boldsymbol{r}^j_n = \hat{\boldsymbol{Y}}^j_n - \boldsymbol{y}^*_n$
18         filter update: $\boldsymbol{Z}^{F,j}_n = \begin{pmatrix}\boldsymbol{X}^{F,j}_n \\ \Theta^{F,m,j}_n\end{pmatrix} = \boldsymbol{Z}^{P,j}_n + K\big(\boldsymbol{r}^j_n + \boldsymbol{\epsilon}^j_n\big)$
19     end
20 end
21 set $\theta_M = \frac{1}{J}\sum_{j=1}^{J}\Theta^{F,M,j}_N$
output: Monte Carlo maximum likelihood estimate, $\theta_M$.
complexity: $\mathcal{O}(MJUN)$
```

Algorithm 7: `iubf()` in **spatPomp**, run as `iubf(P, params` = θ_0, `Nubf` = M, `Nrep_per_param` = $\mathcal{I}$, `Nparam` = N_θ, `nbhd`= $B_{u,n}$, `prop`= p, `cooling.fraction.50` = a, `rw.sd` = $\sigma_{0:N,1:D_\theta}$`)`, using notation from Table 3.1 where `P` is a class '`spatPomp`' object with definitions for `rprocess`, `dunit_measure`, `rinit`, `obs` and `coef`.

```
input: simulator for $f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \mid \boldsymbol{x}_{n-1}\,;\theta)$ and $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$; evaluator for
       $f_{Y_{u,n}|X_{u,n}}(y_{u,n} \mid x_{u,n}\,;\theta)$; data, $\boldsymbol{y}^*_{1:N}$; starting parameter, $\theta_0$; number of replicates per
       parameter, $\mathcal{I}$; number of parameters, $N_\theta$; neighborhood structure, $B_{u,n}$; number of
       iterations, $M$; resampling proportion, $p$; random walk intensities, $\sigma_{0:N,1:D_\theta}$; cooling
       fraction in 50 iterations, $a$.
1  initialize parameters, $\Theta^{F,0,t}_N = \theta_0$
2  for $m$ in $1\!:\!M$ do
3      initialize parameters, $\Theta^{F,m,t}_0 = \Theta^{F,m-1,t}_N$ for $t$ in $1\!:\!N_\theta$
4      initialize filter particles, $\boldsymbol{X}^{F,m,t,i}_0 \sim f_{\boldsymbol{X}_0}\left(\,\cdot\,;\Theta^{F,m,t}_0\right)$ for $t$ in $1\!:\!N_\theta$, for $i$ in $1\!:\!\mathcal{I}$.
5      for $n$ in $1\!:\!N$ do
6          perturb parameters, $\Theta^{P,m,t,i}_n \sim \mathrm{Normal}\big(\Theta^{F,m,t}_{n-1}\,,\ a^{2m/50}\,\Sigma_n\big)$ for $t$ in $1\!:\!N_\theta$, $i$ in $1\!:\!\mathcal{I}$,
            where $\big[\Sigma_n\big]_{d_\theta,d'_\theta} = \sigma^2_{n,d_\theta}\mathbb{1}_{d_\theta,d'_\theta}$
7          proposals, $\boldsymbol{X}^{P,m,t,i}_n \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}\big(\,\cdot \mid \boldsymbol{X}^{F,m,t,i}_{n-1}\,;\Theta^{P,m,t,i}_n\big)$ for $t$ in $1\!:\!N_\theta$, $i$ in $1\!:\!\mathcal{I}$
8
9          $w^{t,i}_{u,n} = f_{Y_{u,n}|X_{u,n}}\big(y^*_{u,n} \mid X^{P,m,t,i}_{u,n}\,;\Theta^{P,m,t,i}_n\big)$ for $u$ in $1\!:\!U$, $t$ in $1\!:\!N_\theta$, $i$ in $1\!:\!\mathcal{I}$
10         $w^{P,t,i}_{u,n} = \prod_{\tilde{n}=1}^{n-1}\Bigg[\prod_{(\tilde{u},\tilde{n})\in B_{u,n}} w^{t,i}_{\tilde{u},\tilde{n}}\Bigg]\prod_{(\tilde{u},n)\in B_{u,n}} w^{t,i}_{\tilde{u},n}$ for $u$ in $1\!:\!U$, $t$ in $1\!:\!N_\theta$, $i$ in $1\!:\!\mathcal{I}$
11         parameter log likelihoods, $r^t_n = \sum_{u=1}^{U}\log\left(\dfrac{\sum_{i=1}^{\mathcal{I}} w^{t,i}_{u,n}\, w^{P,t,i}_{u,n}}{\sum_{\tilde{i}=1}^{\mathcal{I}} w^{P,t,\tilde{i}}_{u,n}}\right)$ for $t$ in $1\!:\!N_\theta$,
12         Select the highest $pN_\theta$ weights: find $s$ with
            $\{s(1),\dots,s(pN_\theta)\} = \{t : \sum_{\tilde{t}=1}^{N_\theta}\mathbf{1}\{r^{\tilde{t}} > r^t\} < pN_\theta\}$
13         Make $1/p$ copies of successful parameters, $\Theta^{F,m,t}_n = \Theta^{F,m,s(\lceil pt\rceil)}_n$ for $t$ in $1\!:\!N_\theta$
14         Set $\boldsymbol{X}^{F,m,t,i}_n = \boldsymbol{X}^{P,m,s(\lceil pt\rceil),i}_n$
15     end
16 end
output: Iterated UBF parameter swarm: $\Theta^{F,M,1:N_\theta}_N$
Monte Carlo maximum likelihood estimate: $\frac{1}{N_\theta}\sum_{t=1}^{N_\theta}\Theta^{F,M,1:N_\theta}_N$.
complexity: $\mathcal{O}(MN_\theta\mathcal{I}UN)$
```

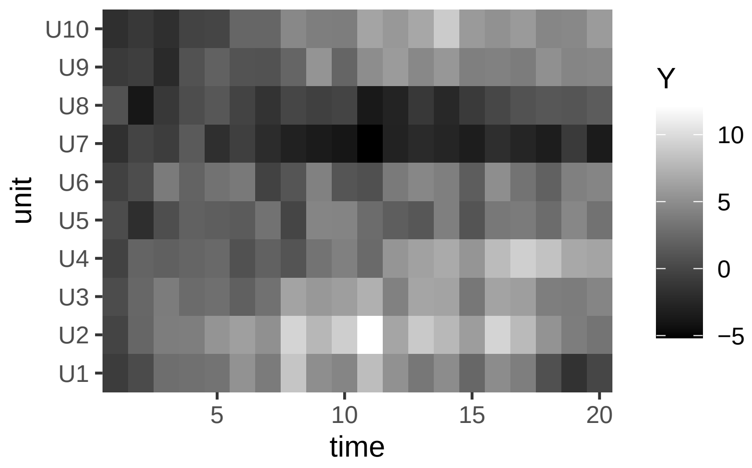


Figure 3.4 – Output of the `plot()` method with argument `type = "h"`.

```
R> bm10 <- bm(U=10, N=20)
R> plot(bm10, nrow = 2)
```

An alternate heat map representation, as shown in Figure 3.4, may also help identify shared patterns among the different units. This can be obtained by supplying the `type = "h"` argument instead of the default `type = "l"`.

```
R> plot(bm10, type = "h")
```

Such plots can help the user qualitatively assess dynamics within and between the units. `plot()` visualizes the results of coercing `bm10` into a class '`data.frame`' object by using the `as.data.frame(bm10)` method. More customized plots can thus be created by using the many plotting options in R for class '`data.frame`' objects. A detailed description of the components of the `bm10` object can be obtained by invoking the `spy()` method from **pomp** as follows (the output is suppressed to conserve space):

```
R> spy(bm10)
```

3.5.1 Computing the likelihood

`bm10` contains all the necessary model components for likelihood estimation using the algorithms discussed in Section 3.3. The standard particle filter, GIRF, ABF, EnKF and BPF can be run to estimate the likelihood of the data at a given parameter set. Here, we use the parameter set that was used to simulate `bm10` and show one likelihood evaluation for each method.

```
R> theta <- coef(bm10)
R> theta
  rho sigma   tau  X1_0  X2_0  X3_0  X4_0  X5_0  X6_0  X7_0  X8_0
  0.4   1.0   1.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
 X9_0 X10_0
  0.0   0.0
R> logLik(pfilter(bm10, params=theta, Np=1000))
[1] -391.7841
R> logLik(girf(bm10, params=theta, Np=100, Nguide=10, Ninter=10, lookahead=1))
[1] -381.0272
R> logLik(abf(bm10, params=theta, Nrep=100, Np=10))
[1] -391.0623
R> logLik(enkf(bm10, params=theta, Np=1000))
[1] -374.0955
R> logLik(bpfilter(bm10, params=theta, Np=1000, block_size=2))
[1] -379.5812
```

We see considerable differences in these initial log likelihood estimates. These might be explained by Monte Carlo variability or bias, and additional investigation is needed to make an assessment.

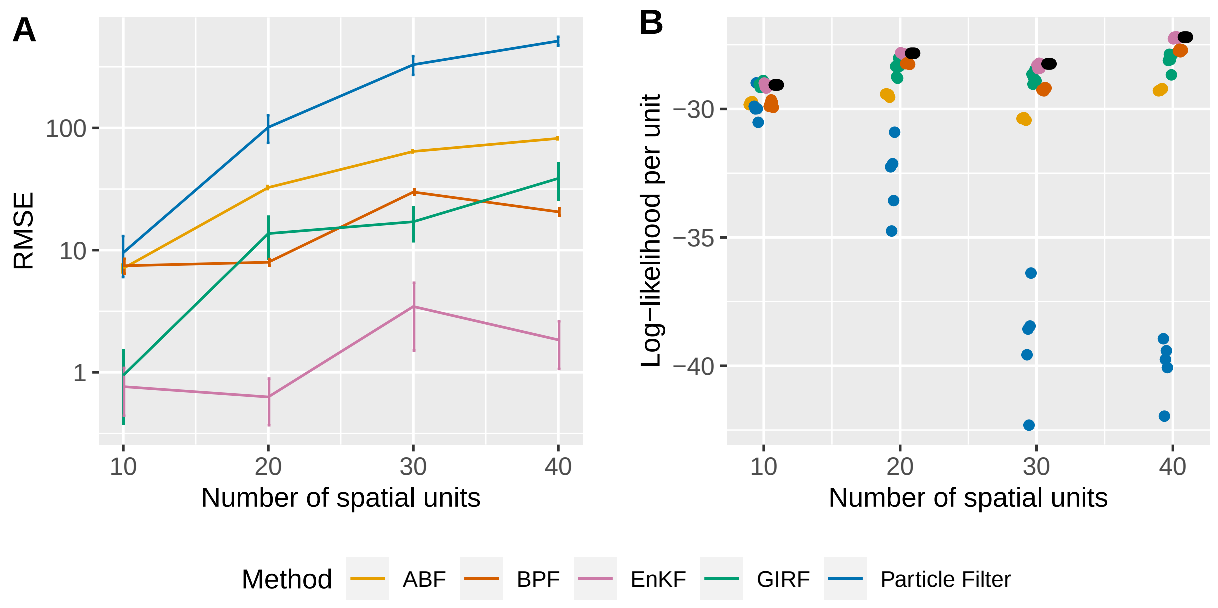


Figure 3.5 – Scaling of various methods on a correlated Brownian motions example. A: RMSE of log likelihood estimates from ABF, BPF, EnKF, GIRF and particle filter on correlated Brownian motions of various dimensions. For a given dimension, we run each method 5 times and calculate the RMSE against the exact log likelihood (obtained via Kalman filter). Error bars represent the variability of the RMSE. B: Log likelihood estimates from ABF, BPF, EnKF, GIRF and particle filter compared to the exact likelihood obtained via Kalman filter (in black).

Both Monte Carlo uncertainty and bias are typical of spatiotemporal filtering applications because of two main factors. First, approximations that give a filtering method scalability have non-negligible impact, primarily bias, on the resulting likelihood estimates. Second, in high-dimensional filtering problems, a Monte Carlo filter can be expected to have non-negligible uncertainty in the likelihood estimate even for methods designed to be scalable. This variability translates to a negative bias in log likelihood estimates, even for methods that are unbiased for the likelihood in the natural scale, due to Jensen's inequality. Overall, all the filters we study have negative bias because they make probabilistic forecasts which involve some approximation to the true forecast distribution under the postulated model. The log likelihood is a proper scoring rule for forecasts, meaning that the exact probabilistic forecast has a higher expected log likelihood than an approximation, if the model is correctly specified (Gneiting et al., 2007).

In practice, we execute multiple runs of each Monte Carlo filtering algorithm to assess the Monte Carlo variance. Bias is harder to assess, except in toy models when a precise likelihood evaluation is computationally tractable.

Figure 3.5 illustrates the result of a more practical exercise of likelihood evaluation. We often start with a model for a small U and evaluate the likelihood many times for each algorithm to quantify the Monte Carlo variability. We then develop our model for increasing U. As U grows the relative performances of the algorithms can vary, so we evaluate the likelihood using all possible methods with several repetitions for a fixed U. On this toy problem with analytically evaluable likelihood, we can compare all likelihood evaluations with the exact likelihood computed using the Kalman filter. As can be seen from Figure 3.5, as the difficulty of the problem increases through the increase in the value of U, we quickly enter the regime in which the particle filter does worse than the methods designed specifically for SpatPOMP models. ABF trades off a slowly growing bias

Table 3.2 – Comparison of the computational resources used by the filtering algorithms.

Method	Resources (core-minutes)	Particles (per replicate)	Replicates	Guide particles	Lookahead
Particle Filter	1.02	2000	-	-	-
ABF	28.67	30	500	-	-
GIRF	111.82	500	-	40	1
EnKF	0.82	1000	-	-	-
BPF	1.06	1000	-	-	-

for a reduced variance by conditioning on a local neighborhood; GIRF reduces variance by using guide functions at intermediate time points to guide prediction particles towards regions of high probability, but this can be computationally costly; BPF approximates the joint filter distribution by resampling independently between blocks of units; EnKF uses a Gaussian-inspired update rule to improve computational efficiency, and on this Gaussian problem the Gaussian approximation made by EnKF is valid, leading to strong performance. In general, since each filtering method has its strengths and limitations, it is worthwhile on a new problem to try them all.

Users will also need to keep in mind considerations about computational resources used up by the available algorithms. Computing resources used by each algorithm for Figure 3.5 are given in Table 3.2. Each algorithm was allowed to use 8 CPU cores to evaluate all the likelihoods and the algorithmic settings were fixed as shown in the table. The time-complexity of GIRF is quadratic in U, due to the intermediate time step loop shown in the pseudocode in Section 3.3.1, whereas the other algorithms scale linearly with U for a fixed algorithmic setting. In addition, GIRF is less scalable than the other filter methods designed for SpatPOMP models. However, a positive feature of GIRF is that it shares with PF the property that it targets the exact likelihood, i.e., it is consistent for the exact log likelihood as the number of particles grows and the Monte Carlo variance apporaches zero. GIRF may be a practical algorithm when the number of units prohibits

PF but permits effective use of GIRF. ABF is implemented such that each bootstrap replicate is run on a CPU core and the results are combined at the end. Since the result from each core is a $U \times N$ matrix, the user should supply more memory if U and/or N are very large. EnKF and BPF generally run the quickest and require the least memory. However, the Gaussian and independent blocks assumptions, respectively, of the two algorithms must be reasonable to obtain likelihood estimates with low bias.

3.5.2 Parameter inference

The correlated Brownian motions example also serves to illustrate parameter inference using IGIRF. Suppose we have data from the correlated 10-dimensional Brownian motions model discussed above. We are interested in estimating the model parameters σ, τ, ρ. The initial conditions, $\{X_u(0), u \in 1\!:\!U\}$, can be considered to be known such that these parameters will not undergo perturbations in IGIRF.

We must construct a starting parameter set for our search.

```
R> start_params <- c(rho = 0.8, sigma = 0.4, tau = 0.2,
+    X1_0 = 0, X2_0 = 0, X3_0 = 0, X4_0 = 0, X5_0 = 0,
+    X6_0 = 0, X7_0 = 0, X8_0 = 0, X9_0 = 0, X10_0 = 0)
```

We can now run `igirf()`. Note that we set the parameter perturbation standard deviation to zero for the initial conditions, which allows us to only estimate our parameters of interest.

```
R> igirf_out <- igirf(
+    bm10,
+    params=start_params,
+    Ngirf=30,
+    Np=1000,
+    Ninter=10,
+    lookahead=1,
+    Nguide=50,
+    rw.sd=rw.sd(rho=0.02, sigma=0.02, tau=0.02,
+                X1_0=0, X2_0=0, X3_0=0, X4_0=0,
+                X5_0=0, X6_0=0, X7_0=0, X8_0=0, X9_0=0, X10_0=0),
+    cooling.type = "geometric",
+    cooling.fraction.50=0.5
+  )
```

The output of `igirf()` is an object of class '`igirfd_spatpomp`'. We can view the final parameter estimate and obtain a likelihood evaluation at this estimate.

```
R> coef(igirf_out)[c('rho','sigma','tau')]
      rho     sigma       tau
0.5560766 0.9642862 1.2031939
R> logLik(igirf_out)
[1] -383.996
```

To get a more accurate likelihood evaluation at the final estimate, the user can run the filtering algorithms with greater computational effort. Since our model is linear and Gaussian, the maximum likelihood estimate of our model and the likelihood at this estimate can be found analytically. The maximum log likelihood is -373.02. An `enkf` run at our `igirf()` parameter estimate yields a log likelihood estimate of -380.91. This shortfall is a reminder that Monte Carlo optimization algorithms should usually be replicated, and may be best used with inference methodology that accommodates Monte Carlo error, as discussed in Section 3.5.3.

A useful diagnostic of the parameter search is the record of improvement of our parameter

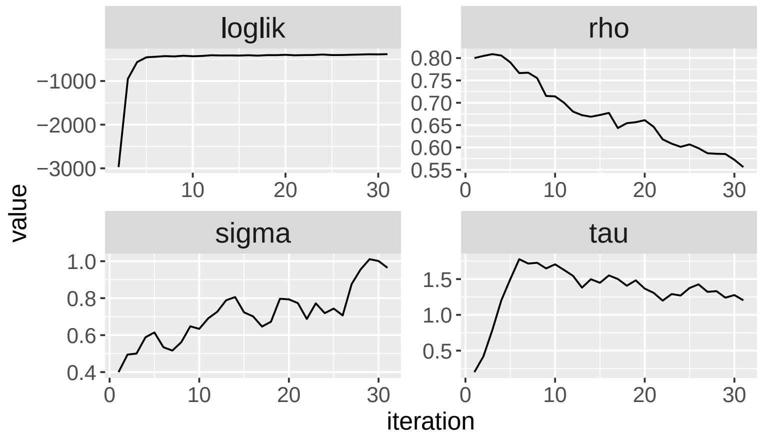


Figure 3.6 – The output of the `plot()` method on an object of class ‘`igirfd_spatPomp`’. This object encodes our model for correlated Brownian motions and the plot produces convergence traces for ρ, σ and τ, and the corresponding log likelihoods. Over 30 iterations `igirf()` has allowed us to get within a neighborhood of the maximum likelihood.

estimates during the course of an `igirf()` run. Each iteration within an `igirf` run provides a parameter estimate and a likelihood evaluation at that estimate. The `plot` method for a class ‘`igirfd_spatPomp`’ object shows the convergence record of parameter estimates and their likelihood evaluations.

```
R> plot(igirf_out, params = c("rho", "sigma", "tau"), ncol = 2)
```

As shown in Figure 3.6, `igirf()` has allowed us to explore the parameter space and climb significantly up the likelihood surface to within a small neighborhood of the maximum likelihood. The run took 5.88 minutes on one CPU core for this example with 10 spatial units. For larger models, one may require starting multiple searches of the parameter space at various starting points by using parallel runs of `igirf()` on a larger machine with multiple cores.

3.5.3 Monte Carlo profiles

Proper interpretation of a parameter estimate requires uncertainty estimates. For instance, we may be interested in estimating confidence intervals for the coupling parameters of spatiotemporal models. These are parameters that influence the strength of the dependence between the latent dynamics in different spatial units. In our correlated Brownian motions example, ρ plays this role. The dependence between any two units is moderated by the distance between the units and the value of ρ.

We can often estimate confidence intervals for parameters like τ and σ which drive the dynamics of each spatial unit. However, coupling parameters can be hard to detect because any signal can be overwhelmed by the inevitably high variance estimates of high-dimensional models. Full-information inference methods like IGIRF which are able to mitigate high variance issues in the filtering step can allow us to extract what limited information is available on coupling parameters like ρ. Here we will construct a profile likelihood for ρ with a 95% confidence interval that adjusts for Monte Carlo error.

A profile over a model parameter is a collection of maximized likelihood evaluations at a range of values of the profiled parameter. For each fixed value of this parameter, we maximize the likelihood over all the other parameters. We often use multiple different starting points for each fixed value of the profiled parameter.

Let us first design our profile over ρ by setting the bounds over all other model parameters from which we will draw starting values for likelihood maximization. It can sometimes be useful to transform the other parameters to an unconstrained scale by using `pomp::partrans()`. For instance, parameters whose natural values are constrained to the non-negative real numbers can be log-transformed to maximize them over the unconstrained real line.

```
R> # center of our hyperbox of starting parameter sets
R> theta <- c("rho" = 0.7, "sigma"=0.7, "tau"=0.6,
+             "X1_0"=0, "X2_0"=0, "X3_0"=0, "X4_0"=0, "X5_0"=0,
+             "X6_0"=0, "X7_0"=0, "X8_0"=0, "X9_0"=0, "X10_0"=0)
R> # set bounds of hyperbox of starting parameter sets for
R> # all non-profiled parameters (use estimation scale to set this)
R> estpars <- setdiff(names(theta),c("rho"))
R> theta_t <- pomp::partrans(bm10,theta,"toEst")
R> theta_t_hi <- theta_t_lo <- theta_t
R> theta_t_lo[estpars] <- theta_t[estpars] - log(2) # lower bound
R> theta_t_hi[estpars] <- theta_t[estpars] + log(2) # upper bound
R> theta_lo <- pomp::partrans(bm10, theta_t_lo, "fromEst")
R> theta_hi <- pomp::partrans(bm10, theta_t_hi, "fromEst")
```

`theta_lo` and `theta_hi` effectively specify a "hyperbox" from which we can draw starting parameter sets for our maximization. Next, we use `pomp::profile_design()` to sample our starting points from this hyperbox. The first argument is the name of the parameter to be profiled over and is set to a range of feasible values for that parameter. The second and third arguments take the hyperbox bounds and the final argument is used to determine how many unique starting positions must be drawn for each value of ρ.

```
R> pomp::profile_design(
+    rho=seq(from=0.2,to=0.5,length=10),
+    lower=theta_lo,
+    upper=theta_hi,
+    nprof=5
+  ) -> pd
```

`pd` is now a class '`data.frame`' object representing random starting positions for our maximizations. Since we 3 starting points for each value of ρ and 10 different values of ρ, we expect 30 rows in `pd`.

```
R> dim(pd)
[1] 30 14
R> head(pd)[c('rho','sigma','tau')]
        rho     sigma       tau
1 0.2000000 0.8949604 0.4172217
2 0.2000000 0.5589302 0.8485934
3 0.2000000 0.9093869 0.9267436
4 0.2333333 0.9045623 0.3579629
5 0.2333333 0.9893592 1.1255378
6 0.2333333 0.8490907 0.8455325
```

We can now run `igirf()` at each starting point. We can run these jobs in parallel using `foreach` and `%dopar%` from the `foreach` package (Wallig and Weston, 2020) and collecting all the results together using `bind_rows` from `dplyr` (Wickham et al., 2020). Once we get a final parameter estimate from each `igirf` run, we can estimate the likelihood at this point by running, say, `enkf()`, 10 times and appropriately averaging the resulting log likelihoods.

```
R>   foreach (
+      p=iter(pd,"row"),
+      .combine=dplyr::bind_rows
+    ) %dopar% {
+      library(spatPomp)
+      igirf_out <- igirf(bm10,
+                         params=p,
+                      Ngirf=bm_prof_ngirf,
+                      Np=1000,
+                      Nguide = 30,
+                      rw.sd=rw.sd(sigma=0.02, tau=0.02),
+                      cooling.type = "geometric",
+                      cooling.fraction.50=0.5)
+
+       ## 10 EnKF log likelihood evaluations
+       ef <- replicate(10,
+                       enkf(igirf_out,
+                            Np = 2000))
+       ll <- sapply(ef,logLik)
+       ## logmeanexp to average log likelihoods
+       ## se=TRUE to estimate Monte Carlo variability
+       ll <- logmeanexp(ll, se = TRUE)
+
+       # Each igirf job returns one row
+       data.frame(
+         as.list(coef(igirf_out)),
+         loglik = ll[1],
+         loglik.se = ll[2]
+       )
+     } -> rho_prof
```

`rho_prof` now contains parameter estimates that result from running `igirf` on each starting parameter in `pd` and the corresponding log likelihood estimates.

```
R> dim(rho_prof)
[1] 30 17
R> print(head(rho_prof)[c("rho","sigma","tau","loglik")], row.names = F)
       rho     sigma       tau    loglik
 0.2000000 1.1821368 0.9660797 -379.4979
 0.2000000 0.8896483 1.2138036 -382.4032
 0.2000000 1.0181503 0.9965749 -378.5496
 0.2333333 1.5835910 0.7610113 -383.6311
 0.2333333 1.0532616 0.8215984 -380.4950
 0.2333333 1.1643568 0.9578773 -377.6221
```

We can can now use the Monte Carlo adjusted profile confidence interval methodology of Ionides et al. (2017) to construct a 95% confidence interval for ρ.

```
R> rho_prof_mcap <- mcap(
+    lp=rho_prof[,"loglik"],
+    parameter=rho_prof[,"rho"]
+  )
R> rho_prof_mcap$ci
[1] 0.2663664 0.4879880
```

The 95% estimated confidence interval for ρ is, therefore, (0.266,0.488). Note that the data in `bm10` are generated from a model with $\rho = 0.4$.

3.6 A spatiotemporal model of measles transmission

A complete **spatPomp** workflow involves roughly two major steps. The first is to obtain data, postulate a class of models that could have generated the data and bring these two pieces together via a call to `spatPomp()`. The second step involves evaluating the likelihood at specific parameter sets and/or maximizing likelihoods under the postulated class of models and/or constructing a Monte Carlo adjusted confidence interval and/or performing a hypothesis test by comparing maximized likelihoods in a constrained region of parameter space with maximized likelihoods in the unconstrained parameter space. We have shown examples of the second major step in a **spatPomp** workflow in Section 5. We now show how to bring our data and models together via a compartment model for coupled measles dynamics in the 5 largest cities in England in the pre-vaccine era.

Compartment models for population dynamics divide up the population into categories (called *compartments*) which are modeled as homogeneous. The rate of flow of individuals between a pair of compartments may depend on the count of individuals in other compartments. Compartment models have widespread scientific applications, especially in the biological and health sciences (Bretó et al., 2009). Spatiotemporal compartment models can be called patch models or metapopulation models in an ecological context, since the full population is divided into a "population" of sub-populations. We develop a spatiotemporal model for disease transmission dynamics of measles within and between multiple cities, based on the model of Park and Ionides (2020) which adds spatial interaction to the compartment model presented by He et al. (2010). We use this example to demonstrate how to construct spatiotemporal compartment models in **spatPomp**. The `measles()` function in **spatPomp** constructs such an object, and here we consider the key steps in this construction. Beyond the examples in the **pomp** and **spatPomp** packages, previous analyses using **pomp** with published open-source code provide additional examples of compartment models (Marino et al., 2019).

3.6.1 Mathematical model for the latent process

Before discussing how to code the model, we first define it mathematically in the time scale $\mathbb{T} = [0, T]$. First we describe how we model the coupling (via travel) between cities. Let $v_{u\tilde{u}}$ denote the number of travelers from city u to $\tilde{u}$. Here, $v_{u\tilde{u}}$ is constructed using the gravity model of Xia et al. (2004):

$$v_{u\tilde{u}} = G \cdot \frac{\overline{\text{dist}}}{\bar{P}^2} \cdot \frac{P_u \cdot P_{\tilde{u}}}{\text{dist}(u, \tilde{u})},$$

where $\text{dist}(u, \tilde{u})$ denotes the distance between city u and city $\tilde{u}$, P_u is the average population for city u across time, $\bar{P}$ is the average population across cities, and $\overline{\text{dist}}$ is the average distance between a randomly chosen pair of cities. In this version of the model, we model $v_{u\tilde{u}}$ as fixed through time and symmetric between any two arbitrary cities, though a natural extension would allow for temporal variation and asymmetric movement between two cities. The gravitation constant G is scaled with respect to $\bar{P}$ and $\overline{\text{dist}}$. The measles model divides the population of each city into susceptible (S), exposed (E), infectious (I), and recovered/removed (R) compartments.

Next we discuss the dynamics within each city, including where the $v_{u\tilde{u}}$ terms eventually feature in (3.2). The latent state process is $\{\boldsymbol{X}(t;\theta), t \in \mathbb{T}\} = \{(X_1(t;\theta), \ldots, X_U(t;\theta)), t \in \mathbb{T}\}$ with $X_u(t;\theta) = (S_u(t;\theta), E_u(t;\theta), I_u(t;\theta), R_u(t;\theta))$. The number of individuals in each compartment for city u at time t are denoted by $S_u(t)$, $E_u(t)$, $I_u(t)$, and $R_u(t)$. The population dynamics are described by the following set of stochastic differential equations:

$$\left.\begin{array}{rcccccc} dS_u(t) & = & dN_{BS,u}(t) & - & dN_{SE,u}(t) & - & dN_{SD,u}(t) \\ dE_u(t) & = & dN_{SE,u}(t) & - & dN_{EI,u}(t) & - & dN_{ED,u}(t) \\ dI_u(t) & = & dN_{EI,u}(t) & - & dN_{IR,u}(t) & - & dN_{ID,u}(t) \end{array}\right\} \text{ for } u = 1, \ldots, U.$$

Here, $N_{SE,u}(t)$, $N_{EI,u}(t)$, and $N_{IR,u}(t)$ denote the cumulative number of transitions, between the compartments identified by the subscripts, up to time t in city u. When these are modeled as integer-valued, the system of differential equations has step function solutions. The recruitment of susceptible individuals into city u is denoted by the counting process $N_{BS,u}(t)$. Here, B denotes a source of individuals that can enter the susceptible population, primarily modeling births. Each compartment also has an outflow, written as a transition to D, primarily representing death, which occurs at a constant per-capita rate μ. The number of recovered individuals $R_u(t)$ in city u is defined implicitly given the known census population $P_u(t) = S_u(t) + E_u(t) + I_u(t) + R_u(t)$. $R_u(t)$ plays no direct role in the dynamics, beyond accounting for individuals not in any of the other classes.

A continuous time latent process model is defined as the limit of the Euler scheme in Box 8 as the Euler time increment approaches zero. We use tildes to distinguish the numerical solution from the continuous time model. The scheme involves initializing numerical analogues $\tilde{S}_u(0)$, $\tilde{E}_u(0)$, $\tilde{I}_u(0)$, $\tilde{R}_u(0) = P_u(0) - \tilde{S}_u(0) - \tilde{E}_u(0) - \tilde{I}_u(0)$ and executing the one-step transitions in the Box at time increments of δ until time T. In the limit as δ approaches zero, this results in a model with infinitesimal mean and variance given by

$$\begin{aligned}
&\mathbb{E}\left[N_{SE,u}(t+dt) - N_{SE,u}(t)\right] \approx \mu_{SE,u}(t) S_u(t) dt + o(dt) \\
&\mathbb{V}\left[N_{SE,u}(t+dt) - N_{SE,u}(t)\right] \approx \left[\mu_{SE,u}(t) S_u(t) + \mu^2_{SE,u}(t) S^2_u(t) \sigma^2_{SE}\right] dt + o(dt), \qquad (3.2) \\
&\qquad \text{where } \mu_{SE,u}(t) = \beta(t) \left[\frac{I_u}{P_u} + \sum_{\tilde{u} \neq u} \frac{v_{u\tilde{u}}}{P_u} \left\{ \frac{I_{\tilde{u}}}{P_{\tilde{u}}} - \frac{I_u}{P_u} \right\} \right].
\end{aligned}$$

We use an integrated noise process with independent Gamma distributed increments that we use to model extrademographic stochasticity on the rate of transition from susceptible classes to exposed

Algorithm 8: Euler increment for our measles model. An Euler increment between times $s_m = m\delta$ to $s_{m+1} = s_m + \delta$ of an Euler scheme whose limit as δ approaches zero is our continuous-time measles latent process model. For notational convenience, Q_1, Q_2, Q_3, Q_4 and Q_5 represent susceptible (S), exposed (E), infectious (I), recovered (R) and natural death (D) statuses, respectively. We keep track of changes to $\tilde{S}_u$, $\tilde{E}_u$, $\tilde{I}_u$ and $\tilde{R}_u$, the numerical analogues of S_u, E_u, I_u and R_u in our mathematical model, by updating each compartment using dynamic rates and our population covariate. The dynamics are coupled via $\mu_{SE,u}$ terms that incorporate travel of infectives from other units. Here, Gamma(α, β) is the gamma distribution with mean $\alpha\beta$ and variance $\alpha\beta^2$. More information about the gamma, multinomial and Poisson distributions can be found in Casella and Berger (1990). The instructions in this box are encoded in the `measles_rprocess` `C` snippet in the following subsection.

```
For $u$ in $1:U$:
    1. Draw unbiased, independent, multiplicative noise for each $\mu_{SE,u}(s_{m+1})$,
       $\Delta\Gamma_{Q_1Q_2,u} \sim \text{Gamma}(\frac{\delta}{\sigma^2_{Q_1Q_2}}, \sigma^2_{Q_1Q_2})$.
       Define $\Delta\Gamma_{Q_iQ_j,u} = \delta$, for $(i,j) \neq (1,2)$
    2. Draw one-step transitions from $\tilde{S}_u(s_m)$, $\tilde{E}_u(s_m)$ and $\tilde{I}_u(s_m)$:
       $\Big(\Delta\tilde{N}_{Q_1Q_2,u}, \Delta\tilde{N}_{Q_1Q_5,u}, \tilde{S}_u(s_m) - \Delta\tilde{N}_{Q_1Q_2,u} - \Delta\tilde{N}_{Q_1Q_5,u}\Big) \sim$
       $\text{Multinomial}\Big(\tilde{S}_u(s_m), p_{Q_1Q_2,u}, p_{Q_1Q_5,u}, 1 - p_{Q_1Q_2,u} - p_{Q_1Q_5,u}\Big)$;
       $\Big(\Delta\tilde{N}_{Q_2Q_3,u}, \Delta\tilde{N}_{Q_2Q_5,u}, \tilde{E}_u(s_m) - \Delta\tilde{N}_{Q_2Q_3,u} - \Delta\tilde{N}_{Q_2Q_5,u}\Big) \sim$
       $\text{Multinomial}\Big(\tilde{E}_u(s_m), p_{Q_2Q_3,u}, p_{Q_2Q_5,u}, 1 - p_{Q_2Q_3,u} - p_{Q_2Q_5,u}\Big)$;
       $\Big(\Delta\tilde{N}_{Q_3Q_4,u}, \Delta\tilde{N}_{Q_3Q_5,u}, \tilde{I}_u(s_m) - \Delta\tilde{N}_{Q_3Q_4,u} - \Delta\tilde{N}_{Q_3Q_5,u}\Big) \sim$
       $\text{Multinomial}\Big(\tilde{I}_u(s_m), p_{Q_3Q_4,u}, p_{Q_3Q_5,u}, 1 - p_{Q_3Q_4,u} - p_{Q_3Q_5,u}\Big)$, where

       $$p_{Q_iQ_j,u} = \frac{(1 - \exp(\sum_k \mu_{Q_iQ_k,u}(s_{m+1})\Delta\Gamma_{Q_iQ_k,u}))\mu_{Q_iQ_j,u}(s_{m+1})\Delta\Gamma_{Q_iQ_j,u}}{\sum_k \mu_{Q_iQ_k,u}(s_{m+1})\Delta\Gamma_{Q_iQ_k,u}}$$

    3. New entries into susceptible class via birth:
       $\Delta\tilde{N}_{BQ_1,u} \sim \text{Poisson}(\mu_{BQ_1,u}(s_{m+1}) \cdot \delta)$
    4. Update compartments by the one-step transitions:
       $\tilde{S}_u(s_{m+1}) = \tilde{S}_u(t_n) - \Delta\tilde{N}_{Q_1Q_2,u} - \Delta\tilde{N}_{Q_1Q_5,u} + \Delta\tilde{N}_{BQ_1,u}$
       $\tilde{E}_u(s_{m+1}) = \tilde{E}_u(t_n) - \Delta\tilde{N}_{Q_2Q_3,u} - \Delta\tilde{N}_{Q_2Q_5,u} + \Delta\tilde{N}_{Q_1Q_2,u}$
       $\tilde{I}_u(s_{m+1}) = \tilde{I}_u(t_n) - \Delta\tilde{N}_{Q_3Q_4,u} - \Delta\tilde{N}_{Q_3Q_5,u} + \Delta\tilde{N}_{Q_2Q_3,u}$
       $\tilde{R}_u(s_{m+1}) = P(s_{m+1}) - \tilde{S}_u(s_{m+1}) - \tilde{E}_u(s_{m+1}) - \tilde{I}_u(s_{m+1})$
```

classes, following Bretó et al. (2009). Extrademographic stochasticity permits overdispersion (McCullagh and Nelder, 1989) which is often appropriate for stochastic models of discrete populations. The '$\approx$' in the above two approximations is a consequence of extrademographic noise, and as σ_{SE} becomes small it approaches equality (Bretó and Ionides, 2011). Here, $\beta(t)$ denotes the seasonal transmission coefficient (He et al., 2010).

3.6.2 Mathematical model for the measurement process

The discrete set of observation times is $t_{1:N} = \{t_n, n = 1, \ldots, N\}$. The observations for city u are bi-weekly new case counts. The observation process $\{\boldsymbol{Y}_n = Y_{1:U,n}, n \in 1\!:\!N\}$ can be written $\{\boldsymbol{Y}_n = \text{cases}_{1:U,n}, n \in 1\!:\!N\}$. We denote the number of true transitions from compartment I to compartment R accumulated between an observation time and some time t before the next observation time to be $C_u(t) = N_{IR,u}(t) - N_{IR,u}(\lfloor t \rfloor)$, where $\lfloor t \rfloor$ is the greatest element of $t_{1:N}$ that is less than t.

Our measurement model assumes that a certain fraction, ρ, called the reporting probability, of the transitions from the infectious compartment to the recovered compartment were, on average, counted as reported cases. Our measurement model is:

$$\text{cases}_{u,n} \,|\, C_u(t_n) = c \sim \text{Normal}(\rho\, c, \rho\,(1-\rho)\, c + (\psi\, \rho\, c)^2),$$

where ψ is an overdispersion parameter that allows us to have measurement variance that is greater than the variance of the binomial distribution with number of trials c and success probability ρ.

3.6.3 Construction of a measles spatPomp object

The construction of class '`spatPomp`' objects is similar to the construction of class '`pomp`' objects discussed by King et al. (2016). Here, we focus on the distinctive features of SpatPOMP models.

Suppose for our example below that we have bi-weekly measles case counts from $U = 5$ cities in England as reported by Dalziel et al. (2016) in the object `measles_cases` of class '`data.frame`'. Each city has about 15 years (391 bi-weeks) of data with no missing data. The first few rows of this data are shown here. We see the column corresponding to time is called `year` and is measured in years (two weeks is equivalent to 0.038 years).

```
    year       city cases
1950.000     LONDON    96
1950.000 BIRMINGHAM   179
1950.000  LIVERPOOL   533
1950.000 MANCHESTER    22
1950.000      LEEDS    17
1950.038     LONDON    60
```

We can construct a `spatPomp` object by supplying three minimal requirements in addition to our data above: the column names corresponding to the spatial units and times of each observation (`'city'` and `'year'` in this case) and the time at which the latent dynamics are supposed to begin. Here we set this to two weeks before the first recorded observations.

```
R> measles5 <- spatPomp(
+    data=measles_cases,
+    units='city',
+    times='year',
+    t0=min(measles_cases$year)-1/26)
```

We can successively add each model component to `measles5` with a call to `spatPomp()` on `measles5` with the argument for each component. To avoid repetition, we will construct all of our model components and supply them all at once in a later call to `spatPomp()`.

First, we consider covariates. Suppose that we have covariate information for each city at each observation time in a class '`data.frame`' object called `measles_covar`. In this case, we have census population and lagged birthrate data. We consider lagged birthrate because we assume children enter the susceptible pool when they are old enough to go to school.

```
    year       city       pop lag_birthrate
 1950.000     LONDON 3389306.0      70571.23
 1950.000 BIRMINGHAM 1117892.5      24117.23
 1950.000  LIVERPOOL  802064.9      19662.96
 1950.000 MANCHESTER  704468.0      15705.46
 1950.000      LEEDS  509658.5      10808.73
 1950.038     LONDON 3388407.4      70265.20
```

If covariate information is not reported at the same frequency as the measurement data, **spatPomp** will linearly interpolate the covariate data, as is the default behavior in **pomp**.

For ease of access, the spatial unit names are mapped to the entries $1, \dots, U$. The mapping for each unit can be found by extracting the unit's position in:

```
R> unit_names(measles)
```

We now move from preparing our covariates to writing our model components. We shall use `C` snippets to specify our model components due to the computational advantages discussed in Section

3.2.5. **spatPomp** compiles the C snippets when building the class '`spatPomp`' object. Before coding up our model components let us specify some global variables in C that will be accessible to all model components. The `globals` argument to a `spatPomp()` call can be used to supply these. A global argument that is automatically created based on the `units` column of our observed data is the `U` variable, which encodes the number of spatial units. Since the movement matrix $\left(v_{u,\tilde{u}}\right)_{u,\tilde{u}\in 1:U}$ is calculable up to the parameter G. We can then define a two-dimensional C array, called `v_by_g` that supplies each $\left(\frac{v_{u,\tilde{u}}}{G}\right)_{u,\tilde{u}\in 1:U}$ in a C snippet called `measles_globals`.

```
R> measles_globals <- Csnippet("
+     const double v_by_g[5][5] = {
+     {0,2.205,0.865,0.836,0.599},
+     {2.205,0,0.665,0.657,0.375},
+     {0.865,0.665,0,1.118,0.378},
+     {0.836,0.657,1.118,0,0.580},
+     {0.599,0.375,0.378,0.580,0}
+     };
+   ")
```

We now construct a C snippet for initializing the latent process at time t_0, which corresponds to `t_0` above. This involves drawing from $f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta)$. The parameter vector θ includes initial proportions of the population in each of the four compartments for each city. The names of these initial value parameters (IVPs) will be passed in alongside other parameters to the `paramnames` argument of the `spatPomp()` constructor with names `S1_0`, ..., `S5_0`, `E1_0`, ..., `E5_0`, `I1_0`, ..., `I5_0` and `R1_0`, ..., `R5_0`. We can use `spatPomp_Csnippet()` to assign the latent states $S_u(0)$, $E_u(0)$, $I_u(0)$, and $R_u(0)$ to the numbers implied by the corresponding IVPs. We do this via the `unit_statenames` argument of `spatPomp_Csnippet()`, which, in our example below, receives the vector `c("S", "E", "I", "R", "C", "W")`. This function recognizes that all $S_u(0), u \in 1\!:\!U$ are stored contiguously in a `C` array (with names `S1`, ..., `S5`) and gives us access to `S1` via `S[0]`. `S2`, ..., `S5` can then be accessed as `S[1]`, ..., `S[U-1]`. Similarly, it provides us access to `E1`, `I1` and `R1` via `E[0]`, `I[0]` and `R[0]`.

The `unit_ivpnames` argument of `spatPomp_Csnippet()` serves a similar purpose. If the user provides `paramnames` to the `spatPomp()` constructor that includes IVP names stored contiguously, their corresponding values are also stored in a `C` array that can be traversed easily. Setting `unit_ivpnames = c("S")` then gives us access to the initial value parameters corresponding to the susceptible class for all units (i.e. `S1_0`, ..., `S5_0`) via `S_0[0]`, ..., `S_0[U-1]`

Finally, the `unit_covarnames` argument of `spatPomp_Csnippet()` similarly allows us to have access to `pop1`, which is the population covariate for our first city, via `pop[0]`. The populations of other cities can then be found via `pop[1]`, ..., `pop[U-1]`

These arguments to `spatPomp_Csnippet()` allow us to have a `code` argument that focuses on specifying the model component. Here, we are able to write a few lines relating the latent states for each city at the initial time to the population in each city and the IVPs.

```
R> measles_rinit <- spatPomp_Csnippet(
+    unit_statenames = c("S", "E", "I", "R", "C", "W"),
+    unit_ivpnames = c("S", "E", "I", "R"),
+    unit_covarnames = c("pop"),
+    code = "
+      for (int u=0; u<U; u++) {
+        S[u] = nearbyint(pop[u]*S_0[u]);
+        E[u] = nearbyint(pop[u]*E_0[u]);
+        I[u] = nearbyint(pop[u]*I_0[u]);
+         R[u] = nearbyint(pop[u]*R_0[u]);
+         W[u] = 0;
+         C[u] = 0;
+       }
+     "
+   )
```

The array variable called `C` above corresponds to $C_u(t)$ defined above. The `W` array variable corresponds to the integrated white noise process with independent gamma increments that helps us model extrademographic stochasticity in the latent process. We will later provide `C` and `W` to the `unit_accumvars` argument of the `spatPomp()` constructor. In **pomp** parlance, `C` and `W` are referred to as accumulator variables because they store changes over the course of a measurement period instead of over the full time scale.

The `rprocess` `C` snippet has to encode only a rule for a single Euler increment from the process model. Further, **spatPomp** provides `C` definitions of all parameters (e.g. `amplitude`) in addition to the state variables and covariates, so the user need only define additional variables used.

```
R> measles_rprocess <- spatPomp_Csnippet(
+    unit_statenames = c("S", "E", "I", "R", "C", "W"),
+    unit_covarnames = c("pop", "lag_birthrate"),
+    code = "
+      double beta, br, seas, foi, dw, births;
+      double rate[6], trans[6];
+      int u,v;
+
+      // school term-time seasonality
+      t = (t-floor(t))*365.25;
+      if ((t>=7&&t<=100) || (t>=115&&t<=199) ||
+            (t>=252&&t<=300) || (t>=308&&t<=356))
+          seas = 1.0+amplitude*0.2411/0.7589;
+        else
+          seas = 1.0-amplitude;
+
+      // transmission rate
+      beta = R0*(gamma+mu)*seas;
+
+      for (u= 0 ; u < U ; u++) {
+        br = lag_birthrate[u];
+
+        // expected force of infection
+        foi = (I[u])/pop[u];
+        for (v=0; v < U ; v++) {
+          if(v != u)
+            foi += g * v_by_g[u][v] * (I[v]/pop[v] -
+                   I[u]/pop[u]) / pop[u];
+        }
+
+        // white noise (extrademographic stochasticity)
+        dw = rgammawn(sigmaSE,dt);
+        rate[0] = beta*foi*dw/dt;  // stochastic force of infection
+        rate[1] = mu;  // natural S death
+        rate[2] = sigma; // rate of ending of latent stage
+        rate[3] = mu; // natural E death
+        rate[4] = gamma; // recovery
+        rate[5] = mu; // natural I death
+
+        // Poisson births
+        births = rpois(br*dt);
+
+        // transitions between classes
+        reulermultinom(2,S[u],&rate[0],dt,&trans[0]);
+        reulermultinom(2,E[u],&rate[2],dt,&trans[2]);
+        reulermultinom(2,I[u],&rate[4],dt,&trans[4]);
+
+        S[u] += births   - trans[0] - trans[1];
+        E[u] += trans[0] - trans[2] - trans[3];
+        I[u] += trans[2] - trans[4] - trans[5];
+        R[u] = pop[u] - S[u] - E[u] - I[u];
+        W[u] += (dw - dt)/sigmaSE;  // standardized i.i.d. white noise
+        C[u] += trans[4];           // true incidence
+      }
+    "
+  )
```

The measurement model is chosen to allow for overdispersion relative to the binomial distribution with success probability ρ. Here, we show the `C` snippet defining the unit measurement model. The `lik` variable is pre-defined and is set to the evaluation of the unit measurement density in either the log or natural scale depending on the value of `give_log`.

```
R> measles_dunit_measure <- spatPomp_Csnippet(
+    code = "
+      double m= rho*C;
+      double v = m*(1.0-rho+psi*psi*m);
+      lik = dnorm(cases,m,sqrt(v),give_log);
+    "
+  )
```

spatPomp will then multiply the unit measurement densities over $u \in 1\!:\!U$ to compute the measurement density at each time. The user may rather directly supply `dmeasure` that returns the product of unit-specific measurement densities. We do so and store the resulting `C` snippet in `measles_dmeasure`, but do not show the code here. This may be used, for instance, to run `pfilter` in **pomp**. We use `Csnippet()` since the argument to the

The `runit_measure` argument of the `spatPomp()` constructor can be supplied a `C` snippet for generating data for a point in time and space given the latent state at that point. This is useful for simulating data from a model. We construct such a `C` snippet here.

```
R> measles_runit_measure <- Csnippet("
+    double cases;
+    double m= rho*C;
+    double v = m*(1.0-rho+psi*psi*m);
+    cases = rnorm(m,sqrt(v));
+    if (cases > 0.0) cases = nearbyint(cases);
+    else cases = 0.0;
+  ")
```

We construct the corresponding `rmeasure` and store it in the `measles_rmeasure` variable. To run the methods EnKF, IEnKF, GIRF and IGIRF, we must supply more specifications about the measurement model. The first two require `eunit_measure` whereas the last two require `skeleton` and additionally `eunit_measure`, `vunit_measure`, `munit_measure` when `kind='moment'` is the desired kind of guide function for GIRF. As was the case with `dunit_measure` and `runit_measure`, the `C` snippets for `eunit_measure`, `vunit_measure` and `munit_measure` can be written assuming that the unit statenames and the `u` and `t` variables have been pre-defined. Within the `C` snippet for `eunit_measure`, a variable named `ey` is defined which should be coded to compute the quantity $\mathbb{E}[Y_{u,n} \,|\, X_{u,n}]$ that `eunit_measure` is tasked to obtain. Similarly, since `vunit_measure` computes a unit measurement variance given the parameter set and the unit states, a variable named `vc` is pre-defined and should take the value of the computed variance. Finally, `munit_measure` returns a moment-matched parameter set given the existing parameter set, the unit states, and an empirically computed variance. Variables with the names of the parameters prefixed by `M_` (e.g. `M_tau`) are pre-defined and assigned to the existing parameter values. The user need only change the parameters that would take on a new value after moment-matching.

For our measles example, `eunit_measure` multiplies the latent modeled cases by the reporting rate.

```
R> measles_eunit_measure <- spatPomp_Csnippet(
+    code = "
+      ey = rho*C;
+    "
+  )
```

`vunit_measure` computes the variance of the unit observation given the unit states and parameter set.

```
R> measles_vunit_measure <- spatPomp_Csnippet(
+    code = "
+      double m = rho*C;
+      vc = m*(1.0-rho+psi*psi*m);
+    "
+   )
```

munit_measure computes a moment-matched size parameter given an empirically calculated variance.

```
R> measles_munit_measure <- spatPomp_Csnippet(
+    code = "
+      double binomial_var;
+      double m;
+      m = rho*C;
+      binomial_var = rho*(1-rho)*C;
+      if(vc > binomial_var) M_psi = sqrt(vc - binomial_var)/m;
+    "
+  )
```

The skeleton model component allows the user to specify a system of differential equations, also called a vector field, which can be numerically solved to evaluate a deterministic trajectory of the latent process at requested times (King et al., 2016). spatPomp_Csnippet() provides an argument called unit_vfnames which provides pointers to vector field values for the corresponding states. The time derivatives for the susceptible classes for our five spatial units, DS1, . . . , DS5 can then be assigned using DS[0], . . . , DS[U-1].

```
R> measles_skel <- spatPomp_Csnippet(
+    unit_statenames = c("S", "E", "I", "R", "C", "W"),
+    unit_vfnames = c("S", "E", "I", "R", "C", "W"),
+    unit_covarnames = c("pop", "lag_birthrate"),
+    code = "
+      double beta, br, seas, foi;
+      int u,v;
+
+       // term-time seasonality
+        t = (t-floor(t))*365.25;
+       if ((t>=7&&t<=100) || (t>=115&&t<=199) ||
+         (t>=252&&t<=300) || (t>=308&&t<=356))
+         seas = 1.0+amplitude*0.2411/0.7589;
+       else
+         seas = 1.0-amplitude;
+
+       // transmission rate
+       beta = R0*(gamma+mu)*seas;
+
+       // deterministic skeleton for each unit
+       for (u = 0 ; u < U ; u++) {
+         br = lag_birthrate[u];
+         foi = I[u]/pop[u];
+         for (v=0; v < U ; v++) {
+           if(v != u)
+             foi+=g*v_by_g[u][v]*(I[v]/pop[v]-I[u]/pop[u])/pop[u];
+         }
+
+         DS[u] = br - (beta*foi + mu)*S[u];
+         DE[u] = beta*foi*S[u] - (sigma+mu)*E[u];
+         DI[u] = sigma*E[u] - (gamma+mu)*I[u];
+         DR[u] = gamma*I[u] - mu*R[u];
+         DW[u] = 0;
+         DC[u] = gamma*I[u];
+       }
+     "
+   )
```

Finally we declare the names of states and parameters. This will allow the compilation of the model components which refer to these names.

```
R> measles_unit_statenames <- c('S','E','I','R','C','W')
R> measles_covarnames <- paste0(rep(c("pop","lag_birthrate"),each=U),1:U)
R> measles_statenames <- paste0(rep(measles_unit_statenames,each=U),1:U)
```

As discussed above, some `unit_statenames` may be used to keep track of accumulations of other `unit_statenames` over an observation time period. The `spatPomp()` constructor provides an argument called `unit_accumvars` to handle this behavior. Among other things, this extends **pomp**'s feature of resetting such variables to zero at the beginning of a measurement period.

A parameter can sometimes be classified as an initial value parameter (IVP) that determines only the initial condition, or a regular parameters (RP) that contributes to the process or measurement model throughout the observed time interval. This classification, when it exists, can be helpful since there are inferential consequences. Precision on estimates of IVPs may not grow with increasing number, N, of observations, whereas for RPs we expect increasing information with increasing N.

```
R> measles_IVPnames <- paste0(measles_statenames[1:(4*U)],"_0")
R> measles_RPnames <- c("R0","amplitude","gamma","sigma","mu",
+    "sigmaSE","rho","psi","g")
R> measles_paramnames <- c(measles_RPnames,measles_IVPnames)
```

The pieces of the SpatPOMP are now combined by a call to `spatPomp`.

```
R> measles5_full <- spatPomp(
+    data = measles5,
+    covar = measles_covar,
+    unit_statenames = measles_unit_statenames,
+    unit_accumvars = c("C","W"),
+    paramnames = measles_paramnames,
+    rinit = measles_rinit,
+    rprocess = euler(measles_rprocess, delta.t=2/365),
+    skeleton=vectorfield(measles_skel),
+    dunit_measure = measles_dunit_measure,
+    eunit_measure = measles_eunit_measure,
+    vunit_measure = measles_vunit_measure,
+    munit_measure = measles_munit_measure,
+    runit_measure = measles_runit_measure,
+    dmeasure = measles_dmeasure,
+    rmeasure = measles_rmeasure,
+    globals = measles_globals
+  )
```

3.6.4 Simulating measles data

Suppose we wanted to simulate data from our model for measles dynamics in the U =5 cities and that we have a parameter set `m5_params` at which we are simulating. We can compare our simulations to the data using the code below and the `plot()` method on the class ‘`spatPomp`’ objects resulting from the simulation and the `measles5_full` object (which includes the true observations) respectively. For epidemiological settings, it helps to set the argument `log=TRUE` of the `plot()` method to focus more on seasonal trends and less on spikes in case counts. The resulting plots are shown in Figure 3.7. This figure may indicate room for improvement in the current parameter vector or model structure. As discussed before, such plots can be customized by working directly with the class ‘`data.frame`’ output of `as.data.frame()`.

```
R> m5_params
       R0 amplitude     gamma     sigma        mu   sigmaSE       rho
 5.68e+01  5.54e-01  3.04e+01  2.89e+01  2.00e-02  2.00e-02  4.88e-01
      psi         g     S1_0      S2_0      S3_0      S4_0      S5_0
 1.16e-01  1.00e+02  2.97e-02  2.97e-02  2.97e-02  2.97e-02  2.97e-02
     E1_0      E2_0      E3_0      E4_0      E5_0      I1_0      I2_0
 5.17e-05  5.17e-05  5.17e-05  5.17e-05  5.17e-05  5.14e-05  5.14e-05
     I3_0      I4_0      I5_0      R1_0      R2_0      R3_0      R4_0
 5.14e-05  5.14e-05  5.14e-05  9.70e-01  9.70e-01  9.70e-01  9.70e-01
     R5_0
 9.70e-01
R> m5_sim <- simulate(measles5_full, params=m5_params)
```

3.7 Conclusion

The **spatPomp** package is designed to be both a tool for data analysis based on SpatPOMP models and a principled computational framework for the ongoing development of inference algorithms. The model specification language provided by **spatPomp** is very general, and implementing a SpatPOMP model in **spatPomp** makes a wide range of inference algorithms available. These two features facilitate objective comparison of alternative models and methods.

As a development platform, **spatPomp** is particularly convenient for implementing algorithms with the plug-and-play property, since models will typically be defined by their `rprocess` simulator,

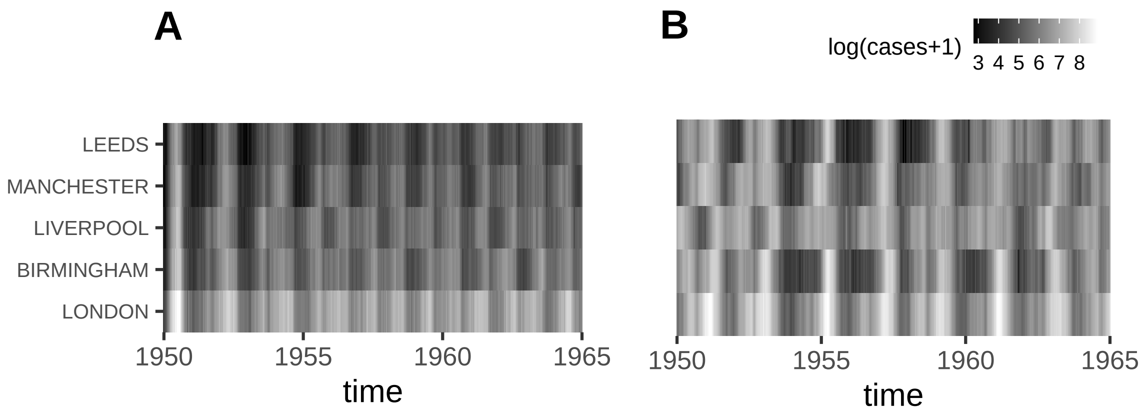


Figure 3.7 – Comparing our measles simulated data to real data. A: Bi-weekly observed measles case counts in the five largest cities in England. B: Simulations from the measles SEIR model encoded in the class `spatPomp` object called `measles5_full`. The figure indicates that the parameter vector and/or the model structure of our SEIR model need to be altered to get patterns similar to those observed in the data.

together with `rmeasure` and often `dunit_measure`, but can accommodate inference methods based on other model components such as `dprocess` if they are available. As an open-source project, the package readily supports expansion, and the authors invite community participation in the **spatPomp** project in the form of additional inference algorithms, improvements and extensions of existing algorithms, additional model/data examples, documentation contributions and improvements, bug reports, and feature requests.

Complex models and large datasets can challenge computational resources. With this in mind, key components of the **spatPomp** package are written in C, and **spatPomp** provides facilities for users to write models either in R or, for the acceleration that typically proves necessary in applications, in C. Multi-processor computing also becomes necessary for ambitious projects. The two most common computationally intensive tasks are the assessment of Monte Carlo variability and the investigation of the roles of starting values and other algorithmic settings on optimization routines. These analyses require only embarrassingly parallel computations and need no special discussion here.

Practical modeling and inference for spatiotemporal partially observed systems, capable of handling scientifically motivated nonlinear, non-stationary stochastic models, is the last open problem of the challenges raised by Bjørnstad and Grenfell (2001). Recent studies have underscored the need for deeper analyses of spatially coupled dynamics (Dalziel et al., 2016), more mechanistic spatial coupling models (Lau et al., 2020), more ways to incorporate covariate information of spatial coupling via cellular data records (Wesolowski et al., 2012, 2015) and more statistical inference methodology that can handle increasing spatial dimension (Lee et al., 2020). The **spatPomp** package addresses these challenges by combining access to modern algorithmic developments with a suitable framework for model specification. The capability to carry out statistically efficient inference for general spatiotemporal systems will promote the development, criticism, refinement and validation of new spatiotemporal models. Nonlinear interacting systems are hard to understand intuitively even when there are relatively few units. Even the single-unit case, corresponding to a low-dimensional nonlinear stochastic dynamic system with a low-dimensional observation process, has rich mathematical theory. Statistically efficient inference for this low-dimensional case was not generally available before the recent development of iterated filtering and particle Markov Chain Monte Carlo methods, and application of these methods has been assisted by their implementations in **pomp**. We anticipate there is much to be gained scientifically by carrying out modeling and inference for spatiotemporal processes with relatively few spatial units but nevertheless surpassing the capabilities of previous software. Facilitating this task is the primary goal of **spatPomp**.

Chapter 4

Analyzing Simulated Measles Data Using Bagged Filters

Data analysis for spatiotemporal systems featuring nonlinear, nonstationary mechanisms and partial observability has been a longstanding open challenge for ecological and epidemiological analysis (Bjørnstad and Grenfell, 2001). A compartment modeling framework for spatiotemporal population dynamics divides the population at each spatial location into categories, called compartments, which are modeled as homogeneous. Spatiotemporal compartment models can be called patch models or metapopulation models in an ecological context. Ensemble Kalman filter (EnKF) methods provide a state-of-the-art approach to inference for metapopulation models (Li et al., 2020) despite concerns that the approximations inherent in the EnKF can be problematic for models that are highly nonlinear or non-Gaussian (Ades and Van Leeuwen, 2015). Our bagged filter methodologies have theoretical guarantees for arbitrarily nonlinear and non-Gaussian models, while having improved scaling properties compared to particle filters.

In this chapter, we put our bagged filters to the test on a nonlinear and non-Gaussian disease model that typifies our problems of interest. We use simulated data for this exercise for two main reasons. First, knowing the model parameters that generated the data allows us to check if our methods will assign higher likelihoods to these parameters. The bias in the likelihood estimates from our methods could conceivably lead to bias in a parameter estimation procedure that is based on our methods. Second, our new methods have tuning parameters (e.g. the size of the space-time neighborhood) whose impact on our methods merits understanding. Knowing the true mechanism that generated the data allows us to reason about how our methods are trying to reconcile the model with the data without worrying about whether the model is misspecified.

4.1 Mathematical models for measles transmission and case data

4.1.1 Mathematical model for the latent process

We consider a spatiotemporal model for disease transmission dynamics of measles within and between multiple cities, based on the model of Park and Ionides (2020) which adds spatial interaction to the compartment model presented by He et al. (2010). The model compartmentalizes the population of each city into susceptible (S), exposed (E), infectious (I), and recovered/removed (R) categories. The number of individuals in each compartment city u at time t are denoted by integer-valued random variables $S_u(t)$, $E_u(t)$, $I_u(t)$, and $R_u(t)$. The population dynamics are written in terms of counting processes $N_{\bullet\bullet,u}(t)$ enumerating cumulative transitions in city u, up to time t, between compartments identified by the subscripts. We model the U largest cities in the UK, ordered in decreasing size so that $u = 1$ corresponds to London. We vary U to test methodologies on a hierarchy of filtering challenges. Our model is described by the following system of stochastic differential equations, for $u = 1, \dots, U$,

$$\begin{aligned} dS_u(t) &= dN_{BS,u}(t) - dN_{SE,u}(t) - dN_{SD,u}(t) \\ dE_u(t) &= dN_{SE,u}(t) - dN_{EI,u}(t) - dN_{ED,u}(t) \\ dI_u(t) &= dN_{EI,u}(t) - dN_{IR,u}(t) - dN_{ID,u}(t) \end{aligned}$$

Here, $N_{BS,u}(t)$ models recruitment into the susceptible population, and $N_{\bullet D,u}(t)$ models emigration and death. The total population $P_u(t) = S_u(t) + E_u(t) + I_u(t) + R_u(t)$ is calculated by smoothing census data and is treated as known. The number of recovered individuals $R_u(t)$ in city u is therefore defined implicitly. $N_{SE,u}(t)$ is modeled as negative binomial death processes (Bretó et al., 2009; Bretó and Ionides, 2011) with over-dispersion parameter σ_{SE}, and rate given by

$$\begin{aligned} \mathbb{E}\big[N_{SE,u}(t+dt) - N_{SE,u}(t)\big] &= \beta(t)\, S_u(t)\Big[\Big(\frac{I_u + \iota}{P_u}\Big) \\ &+ \sum_{\tilde{u} \neq u} \frac{v_{u\tilde{u}}}{P_u}\Big\{\Big(\frac{I_{\tilde{u}}}{P_{\tilde{u}}}\Big) - \Big(\frac{I_u}{P_u}\Big)\Big\}\Big]dt + o(dt), \end{aligned} \tag{4.1}$$

where $\beta(t)$ models seasonality driven by high contact rates between children at school, described by

$$\beta(t) = \begin{cases} \left(1 + a(1-p)p^{-1}\right)\bar{\beta} & \text{during school term,} \\ (1-a)\,\bar{\beta} & \text{during vacation} \end{cases}$$

with $p = 0.759$ being the proportion of the year taken up by the school terms, $\bar{\beta}$ is the mean transmission rate, and a measures the reduction of transmission during school holidays. In (4.1), α is a mixing exponent modeling inhomogeneous contact rates within a city, and ι models immigration of infected individuals which is appropriate when analyzing a subset of cities that cannot be treated as a closed system. The number of travelers from city u to $\tilde{u}$ is denoted by $v_{u\tilde{u}}$. Here, $v_{u\tilde{u}}$ is constructed using a gravity model inspired by that of Xia et al. (2004),

$$v_{u\tilde{u}} = G \cdot \frac{\overline{\text{dist}}}{\bar{P}^2} \cdot \frac{P_u \cdot P_{\tilde{u}}}{\text{dist}(u, \tilde{u})},$$

where $\text{dist}(u, \tilde{u})$ denotes the distance between city u and city $\tilde{u}$, P_u is the average population for city u across time, $\bar{P}$ is the average population across cities, and $\overline{\text{dist}}$ is the average distance between a randomly chosen pair of cities. Here, we model $v_{u\tilde{u}}$ as fixed through time and symmetric between any two arbitrary cities, though a natural extension would allow for temporal variation and asymmetric movement between two cities. The transition processes $N_{EI,u}(t)$, $N_{IR,u}(t)$ and $N_{\bullet D,u}(t)$ are modeled as conditional Poisson processes with per-capita rates μ_{EI}, μ_{IR} and $\mu_{\bullet D}$ respectively, and we fix $\mu_{\bullet D} = 50 \text{ year}^{-1}$. The birth process $N_{BS,u}(t)$ is an inhomogeneous Poisson processes with rate $\mu_{BS,u}(t)$, given by interpolated census data.

4.1.2 Mathematical model for the measurement process

To complete the model specification, we must describe the measurement process. Let $Z_{u,n} = N_{IR,u}(t_n) - N_{IR,u}(t_{n-1})$ be the number of removed infected individuals in the nth reporting interval. Suppose that cases are quarantined once they are identified, so that reported cases comprise a fraction ρ of these removal events. The case report $y^*_{u,n}$ is modeled as a realization of a discretized conditionally Gaussian random variable $Y_{u,n}$, defined for $y > 0$ via

$$\begin{aligned}\mathbb{P}[Y_{u,n}{=}y \mid Z_{u,n}{=}z] \;&=\; \Phi(y + 0.5; \rho z, \rho(1-\rho)z + \psi^2\rho^2 z^2) \\ &\quad -\Phi(y - 0.5; \rho z, \rho(1-\rho)z + \psi^2\rho^2 z^2) \qquad (4.2)\end{aligned}$$

where $\Phi(\cdot;\mu,\sigma^2)$ is the Normal(μ,σ^2) cumulative distribution function, and ψ models overdispersion relative to the binomial distribution. For $y = 0$, we replace $y - 0.5$ by $-\infty$ in (4.2).

This model includes many features that have been proposed to be relevant for understanding measles transmission dynamics (He et al., 2010). Our plug-and-play methodology permits consideration of all these features, and readily extends to the investigation of further variations. Likelihood-based inference via plug-and-play methodology therefore provides a framework for evaluating which features of a dynamical model are critical for explaining the data (King et al., 2008). By contrast, Xia et al. (2004) developed a linearization for a specific spatiotemporal measles model which is numerically convenient but not readily adaptable to assess alternative model choices. Fig. 4.1 shows a simulation from our model, showing that trajectories from this model can capture some features of the system that have been hard to understand: how can it be that disease transmission dynamics between locations have important levels of interaction yet are not locked in synchrony (Becker et al., 2020)? The development of new methods and software allows us to investigate coupled dynamic models like this one in a scientific context. For the moment, we take our model for granted and test our methods on data that are simulated from it.

4.2 Scaling with dimension

We first assess the scaling properties of the filters on the measles model by evaluating the likelihood over varying numbers of units, U, for fixed parameters.

Table 4.1 gives the model parameter values and Table 4.2 gives the algorithmic settings used for the filters. The times in Table 4.2 give the total time required by each algorithm to calculate all its results for Fig. 4.2 using 36 cores. The expected forecast function $\mu(x,s,t)$ needed for ABF-IR and

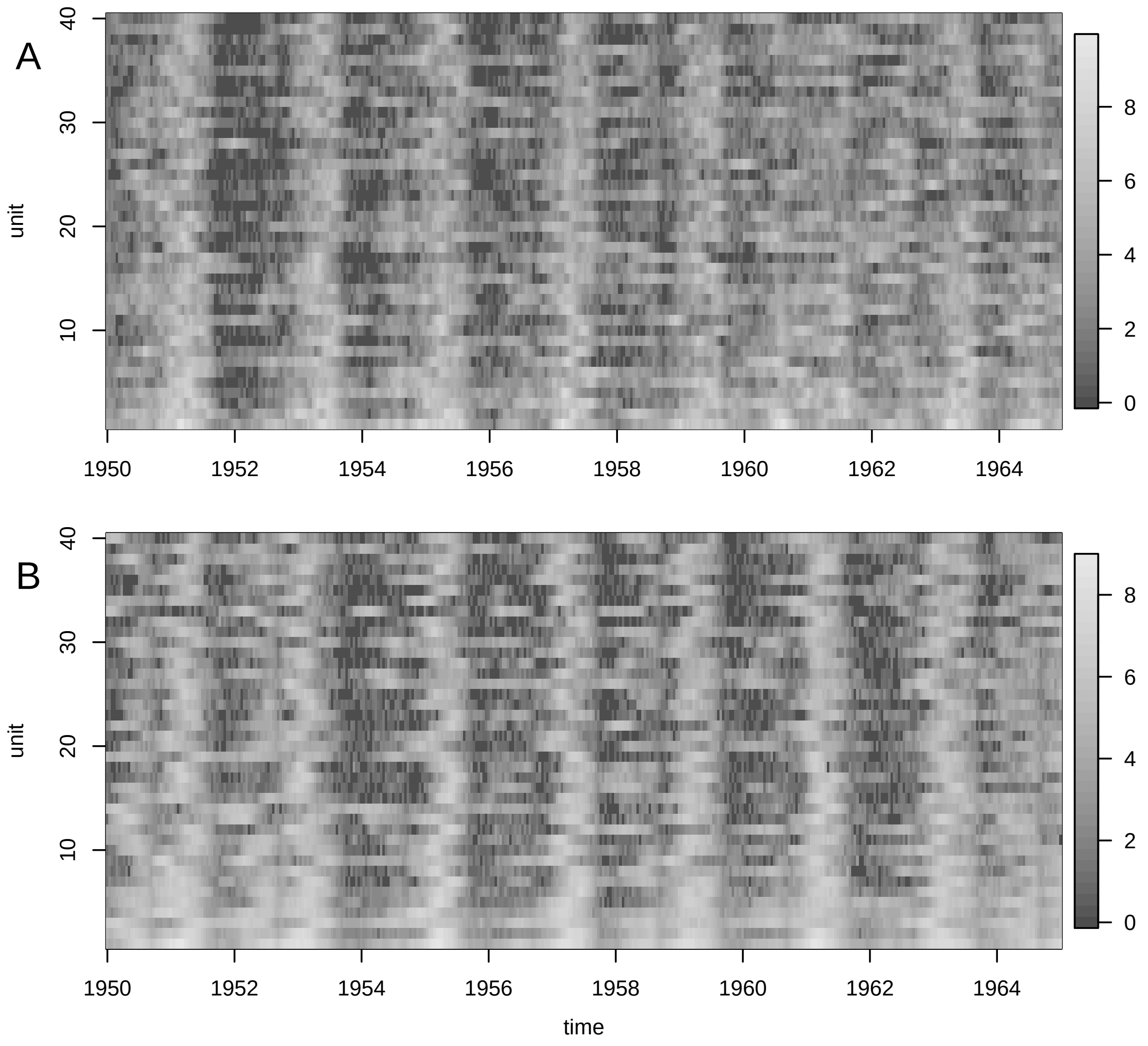


Figure 4.1 – Comparing Log(reported cases +1) for measles simulated data to real data. (A) The measles simulation used for the likelihood slice; (B) The corresponding UK measles data. The simulation shares the biennial pattern, with most but not all cities locked in phase most of the time.

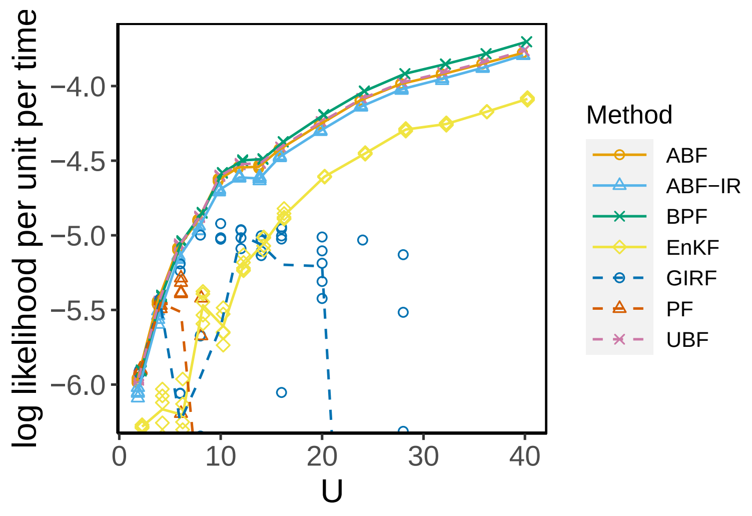


Figure 4.2 – Log likelihood estimates for simulated data of various dimensions. UBF, ABF and ABF-IR are compared with a guided intermediate resampling filter (GIRF), a standard particle filter (PF), a block particle filter (BPF) and an ensemble Kalman filter (EnKF).

parameter	value	unit	description
$\bar{\beta}$	1560.6	year^{-1}	mean contact rate
$\mu_{\bullet D}^{-1}$	50.0	year	mean duration in the population
μ_{EI}^{-1}	7.0	day	latent period
μ_{IR}^{-1}	7.0	day	infectious period
σ_{SE}	0.150	$\text{year}^{1/2}$	process noise
a	0.500	—	amplitude of seasonality
α	1	—	mixing exponent
τ	4	year	delay from birth to entry into susceptibles
ρ	0.5	—	reporting probability
ψ	0.15	—	reporting overdispersion
G	400	—	gravitation constant
$S_u(0), u \in 1:U$	0.032	—	initial susceptible fraction
$E_u(0), u \in 1:U$	0.00005	—	initial exposed fraction
$I_u(0), u \in 1:U$	0.00004	—	initial infectious fraction

Table 4.1 – Parameters for the spatiotemporal measles transmission model

GIRF was computed using a numerical solution to the deterministic skeleton of the stochastic model, i.e, a system of ODEs with derivative matching the infinitesimal mean function of the stochastic dynamic model. In the specifications of $h_u(x)$, $\overleftarrow{\mathrm{v}}_u(V, x, \theta)$ and $\overrightarrow{\mathrm{v}}_u(\theta, x)$, the latent process value x contains a variable C giving the cumulative removed infections in the current observation interval.

In Table 4.2, we see that the effort allocated to UBF, ABF and PF scales linearly with U, since the number of bootstrap replications and particles is fixed in this experiment. GIRF computational effort scales quadratically in U. Its effort is dominated by the guide simulations (which are linear in U) for each observation interval and within each observation interval, there are additional U-dimensional simulations in the intermediate time steps. The effort allocated to ABF-IR scales also with U^2. ABF-IR is more parsimonious with guide simulations (all particles in one bootstrap replication share the same guide simulations) and so the intermediate timestep calculations dominate the effort. To obtain stable variance in the log likelihood estimate, the number of particles and bootstrap replications would have to grow with U. However, given a constraint on total computational resources, the number of particles and bootstrap replications would have to shrink as U increases. The limit studied in this experiment is a balance between the two: the assumption is that one is prepared to invest a growing amount of computational effort as the data grow, but this should not grow too fast. ABF-IR was permitted the greatest computational effort, but the following two considerations balance this:

1. Parallelization. UBF, ABF and ABF-IR are trivially parallelizable. The value of parallelization depends, among other things, on how many replications are being computed simultaneously and on how many cores are available. Nevertheless, it is helpful that the core minute effort requirement for ABF and ABF-IR can be divided by the number of available cores

to give the computational time. Parallelizations of GIRF and PF can be constructed (Park and Ionides, 2020) but these involve non-trivial interaction between processors leading to additional algorithmic complexity and computational overhead.

2. Memory. The intermediate timestep calculations in ABF-IR and GIRF do not add to the memory requirement, and the memory demands of UBF, ABF and ABF-IR are distributed across the parallel computations. A basic PF implementation for a large model can become constrained by its memory requirement (linear in the number of particles) before it can match the processor effort employed by the other algorithms.

In Fig. 4.2, the log likelihood per unit per time increases with U because city size decreases with U. Smaller cities have fewer measles cases, resulting in a narrower and taller probability density function. Fig. 4.2 shows a rapid decline in the performance of the particle filter (PF) beyond $U = 4$. This is a challenging filtering problem, with dynamics including local fadeouts and high stochasticity in each city stabilized at the metapopulation level by the coupling. In this example, GIRF performs poorly suggesting that the simulated moment guide function is less than successful. We used the general-purpose implementation of GIRF in the `spatPomp` package, and there might be room for improvement by developing a model-specific guide function. ABF-IR uses the same guide function, and this may explain why ABF-IR performs worse than ABF here, though ABF-IR is much less sensitive than GIRF to the quality of the guide. ABF and UBF are competing with BPF as winners on this challenge. The bagged filters and BPF have substantial advantages compared to EnKF, amounting to more than 0.2 log likelihood units per observation. We suspect that the limitations of EnKF on this problem are due to the nonlinearity, non-Gaussianity, and discreteness of fadeout and reintroduction dynamics.

	UBF	ABF	ABF-IR	GIRF	EnKF	PF	BPF
particles, J	1	500	200	2000	10000	100000	20000
replicates, $\mathcal{I}$	20000	500	200	—	—	—	—
guide simulations, K	—	—	—	40	—	—	—
lookahead lag, L	—	—	—	1	—	—	—
intermediate steps, S	—	—	$U/2$	U	—	—	—
neighborhood, $B_{u,n}$ or block size	$\{(u,n-1),(u,n-2)\}$			—	—	—	2
measurement mean, $h_{u,n}(x)$	—	—	ρC			—	—
$V = \overrightarrow{\mathrm{v}}_{u,n}(\psi,\rho,x)$	—	—	$\rho(1-\rho)C + \rho^2 C^2 \psi^2$			—	—
forecast mean, $\boldsymbol{\mu}(\boldsymbol{x},s,t)$	—	—	ODE model		—	—	—
$\psi = \overleftarrow{\mathrm{v}}_{u,n}(V,x)$	—	—	$\frac{\sqrt{V-\rho(1-\rho)C}}{\rho C}$		—	—	—
effort (core mins, $U=2$)	28.4	11.5	6.4	2.6	0.3	3.2	0.8
effort (core mins, $U=4$)	35.7	19.0	19.0	5.5	0.6	5.9	1.5
effort (core mins, $U=8$)	52.5	35.0	60.1	12.7	1.2	11.5	2.9
effort (core mins, $U=16$)	87.9	66.7	217.8	36.3	2.4	22.5	5.8
effort (core mins, $U=32$)	155.2	133.3	1032.1	133.7	4.7	45.3	11.6

Table 4.2 – Algorithmic settings for the measles example calculations in Figures 4.2 and 4.3. Computational effort is measured in core minutes for running one filter, corresponding to a point on Figure 4.2. The time taken for computing a single point using the parallel UBF, ABF and ABF-IR implementations is the effort divided by the number of cores, here 36. The time taken for computing a single point using the single core GIRF, EnKF, PF and BPF implementations is equal to the effort in core minutes.

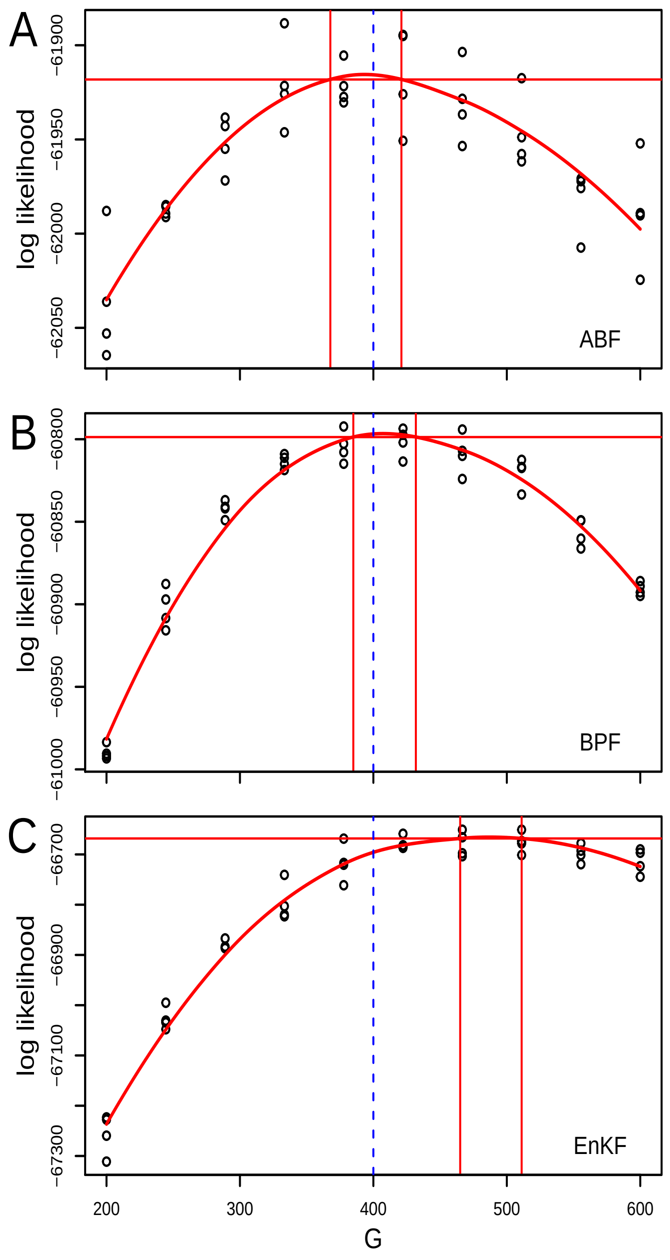


Figure 4.3 – Likelihood slices over the coupling parameter for measles. Likelihoods for the measles model with $U = 40$ cities are computed via (A) ABF; (B) BPF; (C) EnKF. The solid perpendicular lines construct 95% Monte Carlo adjusted confidence intervals (Ionides et al., 2017). The true parameter value is identified by a blue dashed line.

Fig. 4.3(A) demonstrates an application of ABF to the task of computing a slice of the likelihood function over the coupling parameter, G, for simulated data with $U = 40$. This slice varies G while fixing the other parameters at the values used for the simulation. Fig. 4.3(B) shows a similar plot calculated using BPF with comparable computational effort. Both ABF and BPF are successful here, though BPF is more computationally efficient. By contrast, Fig. 4.3(C) shows that EnKF has substantial bias in estimating G, as well as considerably lower likelihood. Likelihood slices have less inferential value than likelihood profiles, but provide a computationally and conceptually simpler setting that can be insightful. Scientifically, the slices in Fig. 4.3 give an upper bound on the identifiability of G from such data, since the likelihood slice provides statistically efficient inference when all other parameters are known.

All the algorithms have various tuning parameters that could influence the results. Generalizable conclusions are hard to infer from numerical comparisons of complex algorithms on complex models. Experimentation with different methods, and their tuning parameters, is recommended when investigating a new model. Some investigations of alternatives are presented below.

4.3 Varying tuning parameters

4.3.1 Varying the neighborhood tuning parameter

We compared five different neighborhoods for the measles model:

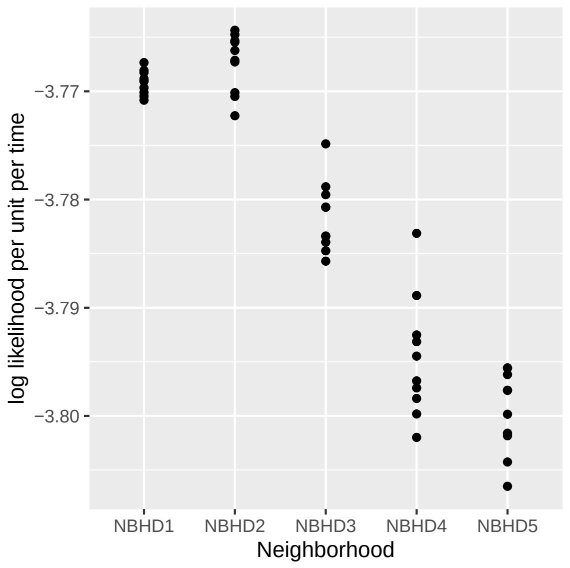


Figure 4.4 – Effect of varying neighborhoods on ABF likelihood estimates. Log likelihood estimates for simulated data from the measles model using ABF, with varying neighborhoods.

NBHD1	One co-located lag	$\{(u, n-1)\}$
NBHD2	Two co-located lags	$\{(u, n-1), (u, n-2)\}$
NBHD3	Three co-located lags	$\{(u, n-1), (u, n-2), (u, n-3)\}$
NBHD4	Two co-located lags and the previous city	$\{(u, n-1), (u, n-2), (u-1, n)\}$
NBHD5	Two co-located lags and London	$\{(u, n-1), (u, n-2), (1, n)\}$

We filtered simulated data for $U = 40$ and $N = 130$, with 10 replications. We used ABF with 200 particles on each of 1000 bootstrap replications. The results are shown in Fig. 4.4. Larger neighborhoods should increase the expected likelihood, but their increased Monte Carlo variability can decrease the expected log likelihood due to Jensen's inequality. In this case, we see that a neighborhood of two co-located lags provides a reasonable bias-variance tradeoff. The time taken for the above calculation was insensitive to the size of the neighborhood. The total run time for each neighborhood in Fig. 4.4 was 223.0 mins for NBHD1, 225.5 mins for NBHD2, 225.4 mins for

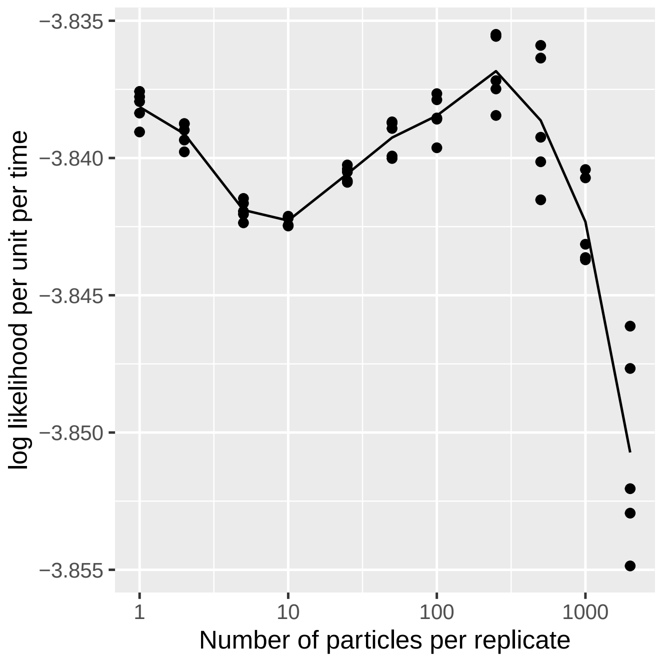


Figure 4.5 – Effect of varying replicates and particles for ABF likelihood estimation. Log likelihood estimates for simulated data from the measles model are computed using ABF, with varying number of replicates and particles.

NBHD3, 218.6 mins for NBHD4, 234.8 mins for NBHD5.

4.3.2 Replicates versus particles

It is necessary in practice to decide whether computational resources are best directed toward a large number of replicates, $\mathcal{I}$, or a large number of particles per replicate, J. Computational effort for ABF and ABF-IR is approximately proportional to $J\mathcal{I}$, and UBF corresponds to ABF with $J = 1$. Suitable algorithmic parameters may depend on the model under consideration, and here we consider resource allocation for the measles model above using simulated data with $U = 40$ and $N = 104$. From Figure 4.2, we know that this implementation of the model is well suited to ABF and UBF. ABF-IR performs less well, and the weak performance of GIRF suggests that the weakness may be due to an inadequate guide function. Figure 4.5 investigates the tradeoff between

UBF and ABF by plotting evaluated log likelihood against $\mathcal{I}$ with J chosen to give approximately a constant computational effort. Algorithmic settings and run times are reported in Table 4.3. For our implementation, choosing J very low and $\mathcal{I}$ correspondingly high led to greater computation time, perhaps because the code was written to parallelize nicely when J is relatively large.

We interpret the bimodal curve as follows. When ABF is carried out with an inadequate number of particles for each bootstrap replicate, the algorithm cannot make a good representation of a draw from the adapted distribution. In that case, there is an advantage to using UBF, which does not attempt to carry out adapted simulation. For very small numbers of particles per replicate, the ABF algorithm behaves like a not-quite-properly-weighted version of UBF. Large numbers of particles per replicate presumably lead to improved Monte Carlo representation of draws from the adapted distribution, but computational cost constraints prevent combining this with a large number of replicates. We see a mode around 500 particles per replicate where ABF out-performs UBF. On this problem, UBF is relatively successful, presumably because the measles dynamics in each city are strongly attracted toward relatively few stable cycles (annual epidemics, or peaks in odd years, or peaks in even years) and a tractable number of simulations can represent all these scenarios. The Lorenz model of Sec. 2.4.2 provides an alternative situation, where adaptation has more advantages. Also, the plug-and-play guide function appears to operate successfully in the Lorenz model, as evidenced by relatively strong performance from ABF-IR and GIRF.

One may observe the dip between the two modes in Figure 4.5 and ask how one may diagnose if one is in that situation. First, the bimodal pattern seen here applies to the specific measles model and neighborhood structure of ABF and UBF applied in this example. Our general advice for likelihood estimation is to start with a moderately large neighborhood (with points in space-time with the same time component and points from previous times) and a large enough Monte Carlo

J	$\mathcal{I}$	time
2000	500	70.7
1000	1000	71.3
500	2000	72.4
250	4000	75.0
100	10000	80.9
50	20000	93.8
25	40000	114.9
10	40000	71.4
5	40000	56.1
2	40000	47.7
1	40000	43.0

Table 4.3 – Bootstrap replications and particles per replicate for Figure 4.5.

effort ($\mathcal{I}$ and J) to minimize bias. The user can then pare down the neighborhood to understand what minimal neighborhood structure is required for the problem at hand. Second, the function of $\mathcal{I}$ is to reduce the variance in our likelihood estimates whereas J controls the quality of our adapted simulations. The main lesson from Figure 4.5 is that both functions must be done well if we choose $J > 1$. The surefire way to do so is to ramp up both values in our exploratory analysis and record our results. We can then gradually decrease both until we are satisfied with both our computational investment and our estimates' bias and variance for the specific problem.

4.4 Likelihood maximization and profile likelihood

Our focus with the bagged filters is on evaluating the likelihood function for SpatPOMP models via filtering. Although the likelihood function is fundamental for inference, evaluation alone is not sufficient. Likelihood maximization enables calculation of the maximum likelihood estimate, profile likelihood confidence intervals, likelihood ratio tests and likelihood-based model selection criteria. Iterated filtering methodology (Ionides et al., 2006, 2011, 2015) provides an approach to

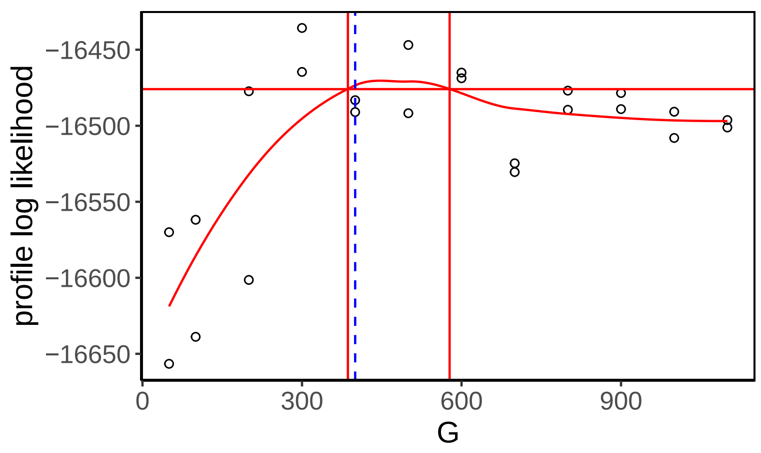


Figure 4.6 – Likelihood profile over coupling parameter using IABF. An iterated bagged filter used to maximize the likelihood, compute a profile likelihood, and hence construct a confidence interval. The profiling is carried out over the coupling parameter, G. We use data for $U = 20$ cities each with $N = 208$ bi-weeks to construct the interval.

extending filtering algorithms to likelihood maximization algorithms. Here, we demonstrate one such extension in the context of bagged filters. The results of applying an iterated bagged filter algorithm on the measles model are shown by constructing a 95% profile confidence interval for the coupling parameter, G, in Fig. 4.6. The algorithm is described next. This demonstration provides a proof of concept to motivate future work.

Iterated filtering approaches apply filtering to a modified version of the model where parameters are perturbed at each time point. The filtering procedure directs the perturbed parameters toward values consistent with the data. At the end of each filtering operation, a parameter updating rule is applied and a new filtering iteration is started with reduced perturbation variance. Under suitable conditions, iterative procedures of this type converge to a neighborhood of a maximum likelihood parameter despite the presence of Monte Carlo filtering error. We implemented an iterated bagged filter procedure, described by the pseudocode below.

The pseudocode is presented for an iterated adapted bagged filter (IABF) but the iterated un-

Table 4.4 – IABF inputs and outputs.

```
input: same as ABF algorithm in Chapter 2 plus
  Number of maximization iterations, $M$
  Number of parameter vectors, $K$
  Perturbation variance, $\Sigma$
  Variance reduction factor after 50 iterations, $\alpha$
  Starting parameters, $\theta^{(0)}_{1:K}$
  Resampling proportion, $p$
output:
  Parameter estimates approaching the maximum likelihood estimate, $\theta^{(M)}_{1:K}$
implicit loop:
  $k$ in $1{:}K$, $i$ in $1{:}\mathcal{I}$
```

IABF. Iterated adapted bagged filter.

```
For $m$ in $1{:}M$
  $\theta^F_{0,1:K} = \theta^{(m-1)}_{1:K}$
  Initialize adapted simulation: $\boldsymbol{X}^F_{0,i,k} \sim f_{\boldsymbol{X}_0}(\boldsymbol{x}_0\,;\theta^F_{0,k})$
  For $n$ in $1{:}N$
    $\theta^P_{n,k} \sim \mathrm{Normal}[\theta^F_{n-1,k}, \alpha^{2m/50}\Sigma]$
    $\boldsymbol{X}^P_{n,i,j,k} \sim f_{\boldsymbol{X}_n|\boldsymbol{X}_{n-1}}(\boldsymbol{x}_n \,|\, \boldsymbol{X}^F_{n-1,i,k}\,;\theta^P_{n,k})$
    Measurement weights: $w^M_{u,n,i,j,k} = f_{Y_{u,n}|X_{u,n}}(y^*_{u,n} \,|\, X^P_{u,n,i,j,k}\,;\theta^P_{n,k})$
    Adapted resampling weights: $w^A_{n,i,j,k} = \prod_{u=1}^{U} w^M_{u,n,i,j,k}$
    State resampling: $\mathbb{P}[r(i,k)=a] = w^A_{n,i,a,k}\Big(\sum_{\xi=1}^{J} w^A_{n,i,\xi,k}\Big)^{-1}$
    $\boldsymbol{X}^A_{n,i,k} = \boldsymbol{X}^P_{n,i,r(i,k),k}$
    $w^P_{u,n,i,j,k} = \prod_{\tilde{n}=1}^{n-1}\Big[\frac{1}{J}\sum_{\xi=1}^{J}\prod_{\tilde{u}:(\tilde{u},\tilde{n})\in B_{u,n}} w^M_{\tilde{u},\tilde{n},i,\xi,k}\Big]\prod_{\tilde{u}:(\tilde{u},n)\in B_{u,n}} w^M_{\tilde{u},n,i,j,k}$
    $\ell^{\mathrm{MC}}_{n,k} = \sum_{u=1}^{U}\log\left(\frac{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^M_{u,n,i,j,k}w^P_{u,n,i,j,k}}{\sum_{i=1}^{\mathcal{I}}\sum_{j=1}^{J} w^P_{u,n,i,j,k}}\right)$
    Select the highest $pK$ likelihoods: find $s$ with
    $\{s(1),\dots,s(\lceil pK\rceil)\} = \{k : \sum_{\tilde{k}=1}^{K}\mathbf{1}\{\ell^{\mathrm{MC}}_{n,\tilde{k}} > \ell^{\mathrm{MC}}_{n,k}\} < (1-p)K\}$
    Make $1/p$ copies of successful parameters, $\theta^F_{n,k} = \theta^P_{n,s(\lceil pk\rceil)}$
    $\boldsymbol{X}^F_{n,i,k} = \boldsymbol{X}^A_{n,i,s(\lceil pk\rceil)}$
  End for
  $\theta^{(m)}_k = \theta^F_{N,k}$
End for
```

adapted filter (IUBF) corresponds to the case $J = 1$ and the iterated bagged filter with intermediate resampling (IABF-IR) follows by adding the intermediate resampling procedure used by ABF-IR. The filtering step of IABF uses ABF to estimate the likelihood at the K perturbed parameter sets. The selection step does not resample parameters based on these estimated likelihood. Rather, it selects the parameters with the top p quantile of likelihoods and copies them appropriately to get K new parameters for the next filtering step. This quantile-based resampling allows us to maintain the diversity of the K parameter sets and avoid *parameter degeneracy*, whereby very few parameters are resampled, leading to an inefficient search of parameter space.

For simplicity, this description assumes that parameters are transformed so that their values are unconstrained. Our software implementation, provided in the R package `spatPomp` (Asfaw et al., 2021b), provides facilities for carrying out such transformations. The Gaussian distribution used for perturbations, and the geometric perturbation variance reduction factor, α, are convenient specifications but are not required in theory (Ionides et al., 2015). As another simplification, the pseudocode for IABF represents the logical structure of the algorithm without attending to issues of memory management and parallelization. For implementation issues, we refer to `spatPomp` (Asfaw et al., 2021b).

In Figure 4.6, we use this IABF implementation to construct a profile likelihood for the measles model. We use $J = 1$, $\mathcal{I} = 30000$, $K = 250$, $p = 0.8$, $\alpha = 0.5$, $M = 15$ and Σ set to be a diagonal matrix with perturbation variance for each non-initial value parameter set to 0.02. For this exercise, we fix the initial value parameters at their true values.

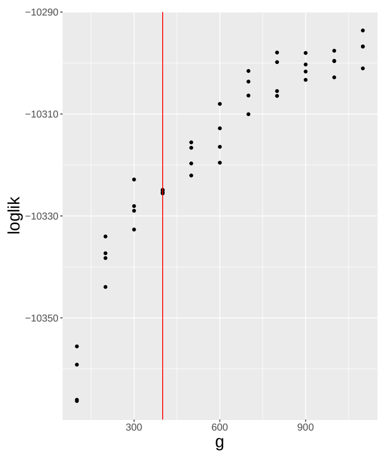
Monte Carlo methods for computing and maximizing the log likelihood suffer from bias and variance, both of which can be considerable for large datasets and complex models. Appropriate inference methodology, such as Monte Carlo adjusted profile (MCAP) confidence intervals, can accommodate substantial Monte Carlo variance so long as the bias is slowly varying across the statistically plausible region of the parameter space (Ionides et al., 2017; Ning et al., 2021). Fig. 4.6 constructs an MCAP 95% confidence interval for the coupling parameter, G, using an iterated unadapted bagged filter to maximize over the parameters, a, $\bar{\beta}$, σ_{SE}, ψ, μ_{EI} and μ_{IR}. This simulation study, carried out with $U = 20$ and $N = 208$, shows that G is identifiable via likelihood-based inference in the absence of assumptions about these parameters.

4.5 Two insights from experiments

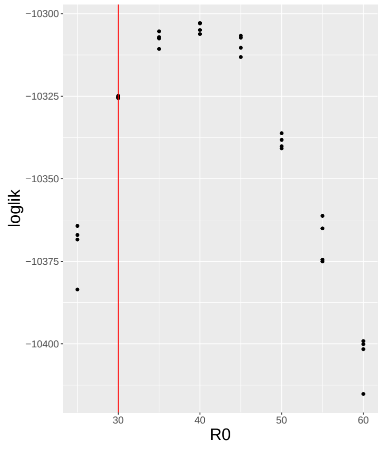
In this section, we show first how inference on subsetted measles data can lead to unexpected results. We then illustrate how the localization bias in likelihood estimation can have nontrivial implications for inference.

4.5.1 Implications of subsetting measles data

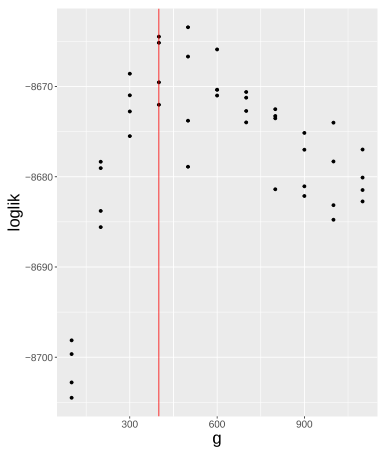
One result that we were not expecting is illustrated in Figure 4.7. Simulated data for 40 cities was generated using the measles model described in Section 4.1. We took a subset of the data for the largest ten cities and constructed an ABF slice of the coupling parameter, G, and the reproductive number parameter, R_0. The values of these parameters used to generate the data were 400 and 30.6 respectively. We were surprised to find that ABF assigned higher likelihoods to values of G and R_0 higher than the values that generated the data. It turns out that subsetting ten cities from data generated by a 40-city model and fitting the data results in much higher coupling and transmissibility parameters. The subsetted data "prefer" greater transmission by having more travelers and higher infectivity. Though we see this phenomenon in a simulation study, we believe



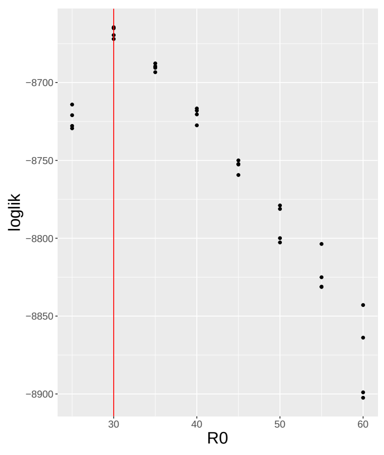
(a)



(b)

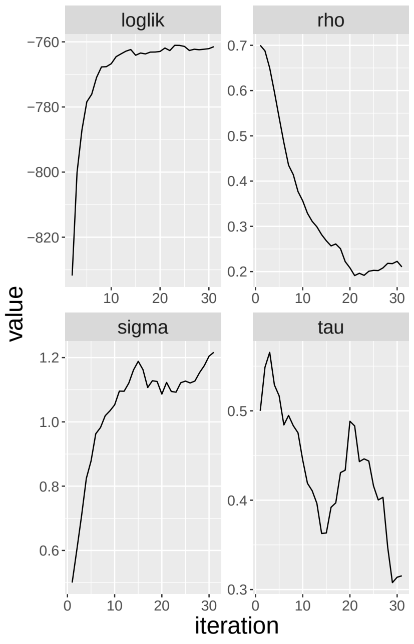


(c)

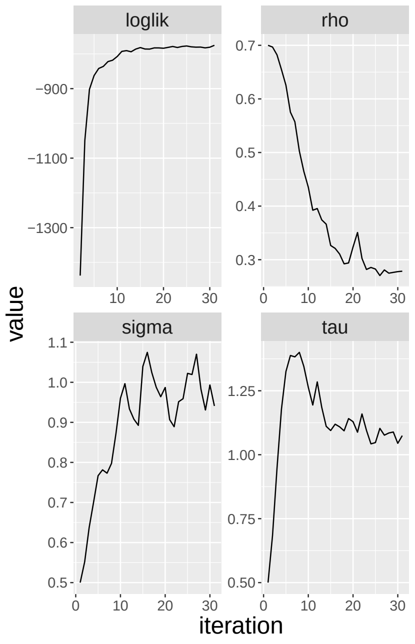


(d)

Figure 4.7 – Illustration of the effect of subsetting data on inference. Subsetting data for 10 cities from a 40-city simulation and evaluating a likelihood slice results in higher likelihood for "false" parameters that imply higher coupling and transmission. (a) and (b) correspond to likelihood slices along the coupling (G) and the transmission (R_0) parameters when 10-city data is subsetted from data generated by a 40-city model. (c) and (d) represent likelihood slices along the same parameters when data is simulated from a 10-city model instead. The true parameter values used to generate the data in both cases are $G = 400$ and $R_0 = 30.6$



(a)



(b)

Figure 4.8 – Illustrating the impact of small neighborhood ABF bias on parameter inference. (a) corresponds to an IABF run using a neighborhood $B_{u,n} = \{(u, n-1), (u, n-2)\}$ whereas (b) corresponds to a run with $B_{u,n} = \{(u-1, n), \dots, (u-9, n), (u, n-1)\}$. In both cases, the ABF likelihood is increasing throughout the iterations. However, the large bias in likelihood evaluation in (a) results in parameter estimates that have much lower true likelihoods than the parameter estimates from (b). This is illustrated by parameter estimates in (b) that are closer to the parameters that generated the data: $\sigma = 1$, $\tau = 1$ and $\rho = 0.4$.

it could have implications for studies that treat a collection of cities like a closed system.

4.5.2 Implications of ABF bias for inference

In this section, we show how the localization bias in likelihood estimation can have unexpected implications for inference and provide general guidance.

In Figure 4.8 we illustrate the progress of two parameter inference runs using IABF on a 10-dimensional simulated Brownian time series data (please check Section 3.5 for a complete description of this model). The coupling, process noise and measurement noise parameters used to simulate the data were $\rho = 0.4$, $\sigma = 1$ and $\tau = 1$, respectively. For the IABF runs we used $\mathcal{I} = 100$, $J = 40$ and

$K = 100$ for $M = 30$ iterations with starting parameters $\rho = 0.7$, $\sigma = 0.5$ and $\tau = 0.5$. Importantly, for Figure 4.8a we set the neighborhood $B_{u,n} = \{(u, n-1), (u, n-2)\}$ whereas for Figure 4.8b $B_{u,n} = \{(u-1, n), \ldots, (u-9, n), (u, n-1)\}$. Since we are working with a fully Gaussian system, the MLE and its true log likelihood of -733.06 can be found analytically from the data. Figures 4.8a and 4.8b each illustrate the progress of the inference algorithm as the iterations proceeded. The log likelihood plot in each subplot corresponds to an estimate of the log likelihood of the data at the parameter estimates reached at the end of an iteration.

There are a few things to notice. Critically, in the case of Figure 4.8a the measurement noise parameter estimate went *down* from a starting point of 0.5 to a final estimate of about 0.32 while the process noise parameter started at 0.5 and went up to 1.22. The coupling parameter dropped from 0.7 to 0.21. All the while, the ABF likelihood is increasing from about -831.79 at the end of the first iteration to -761.49 at the end of the thirtieth iteration. The true log likelihood of the data at the final parameter estimate can be analytically evaluated to be -794.08 whereas the true log likelihood at the starting parameter set is -1272.52.

On the other hand, in 4.8b, the measurement noise parameter estimate went *up* to 1.08 and the process noise went up to 0.94. The final coupling parameter estimate was 0.28. Here again, the ABF likelihood increased throughout from -1437.90 to -775.40.

Two unexpected observations about the two IABF runs are that estimates of the measurement noise parameter are on opposite sides of the starting point and that the run in Figure 4.8a seems to overestimate the log likelihood throughout the iterations. Both of these outcomes result from the bias in ABF likelihood evaluation. For Figure 4.8a, $B_{u,n}$ was set to a neighborhood in space-time that did not take into account any current observations at time n such that resampling within each bootstrap filter was based on the consistency of the current prediction particle with the current data. In other words, all prediction particles had prediction weights equal to 1. This means that the algorithm does not target the correct distribution when it resamples from the prediction particles. The quantity being maximized using the iterated algorithm (the ABF likelihood) is different from the true likelihood and the outputted parameter set is not the MLE. On the other hand, the run corresponding to Figure 4.8b uses a larger neighborhood in space-time that uses observations from the current time. The prediction weights are now closer to the proper importance weights and the ABF likelihood is less biased for the true likelihood.

This phenomenon can be thought of as a bias-variance trade-off. A smaller space-time neighborhood leads to large bias in the likelihood evaluation whereas a larger neighborhood leads to higher variance of the likelihood estimates (more numbers being multiplied to get prediction weights) but lower bias. This experiment suggests that users of our bagged filter methods should generally start with large enough neighborhoods in space-time that include points with the same time coordinate (i.e. $B_{u,n}$ should include $B_{\tilde{u},n}$ for some $\tilde{u} < u$). This may lead to high variance estimates, but this can be countered with higher Monte Carlo effort. Otherwise, starting with too small of a neighborhood can lead to considerable bias in our parameter estimates.

Chapter 5

Concluding Thoughts

For applied statisticians with an interest in epidemiology, an open area of research is methodology for fitting coupled mechanistic POMP models to disease data. Such models allow the analyst to construct a model that is as faithful to the science of the disease transmission mechanism as possible without losing the ability to criticize and develop them via likelihood-based inference. In this thesis, I have contributed to this area by working on new methodology and new software. Below, I summarize these contributions in greater detail and discuss future directions.

Bagged filters

The bagged filter algorithms intorduced in Chapter 2 use independent filters to approximately sample from the adapted distribution. By avoiding resampling among the bootstrap replicates, the bagged filter algorithms maintain a diversity of regions of the state space that are explored. In a later step, the filter weights are calculated locally in space and time, which allows us to trade off some bias in likelihood evaluation for reduced variance in our likelihood estimates. I presented theoretical results showing that, under some assumptions, the asymptotic variance of the bagged filter likelihood estimator scales polynomially in the number of spatial units being studied. This is a significant improvement over the exponential rate seen in the bootstrap filter (Rebeschini and van Handel, 2015). I also show simulation results that show the bagged filter performing competitively with other existing methods in three different models.

Chapter 4 contains a more extensive study of our methods on a model that exemplifies our target problem: a nonlinear and non-Gaussian measles model for cities and towns in the United Kingdom and Wales. I show that the bagged filters scale favorably with the number of spatial units in the coupled model. I also show how the methods depend on their tuning parameters and give general advice about how a user can decide on such parameters. The inferential promise of the bagged filters is shown in a likelihood slice over the coupling parameter for the model. Inspired by this, an iterated bagged filter algorithm with parameter perturbations is introduced. This algorithm is shown to produce a 95% confidence interval for the coupling parameter that contains the true parameter value in a simulation study with twenty cities and towns. I also show two simulation studies that show the impact of bagged filter bias on inference and the dangers of parameter inference using data that assumes a closed population.

Exploring subsetting bias

An exploration of the hazard associated with fitting mechanistic models that assume a closed population is a possible next step for this research. He et al. (2010) account for importation of infectious cases in an uncoupled model by incorporating a parameter for the mean number of infectives visiting a city at a given time. However, this is approach does not mechanistically link the latent states in different spatial units dynamically. A true data analysis that compares two models where coupling does and does not exist from a subset of towns and cities could help us observe the difference in parameter estimates and the extent to which a fully coupled model fits the data better. This could be done by incorporating a $U \times U$ matrix, κ, that "switches off" coupling in the force of infection equation among some spatial units. The corresponding equation to 4.1 would be:

$$\begin{aligned}\mathbb{E}\big[N_{SE,u}(t+dt) - N_{SE,u}(t)\big] \quad &= \quad \beta(t)\, S_u(t)\Big[\left(\frac{I_u+\iota}{P_u}\right)^{\alpha} \\ &+ \sum_{\tilde{u}\neq u} \kappa_{u\tilde{u}} \frac{v_{u\tilde{u}}}{P_u}\left\{\left(\frac{I_{\tilde{u}}}{P_{\tilde{u}}}\right)^{\alpha} - \left(\frac{I_u}{P_u}\right)^{\alpha}\right\}\Big]dt + o(dt)\end{aligned}$$

We can then run the IABF algorithm introduced in section 4.4 for parameter inference to maximize the likelihoods under two configurations of the κ matrix.

Beyond IABF

IABF shows that the bagged filter methods can be used for parameter inference but is computationally expensive. The algorithm requires on the order of 10000 bootstrap replicates per starting parameter set. A modest exploration of parameter space requires a number of starting parameter sets on the order of 100. Even an implementation of IABF that parallelizes over all the cores of a machine could only complete the profile confidence interval calculation in section 4.4 by occupying thirty nodes of the Great Lakes cluster for two weeks. The computation cost of Figure 4.6 came to about \$3000, an untenable expense. Further, we expect the number of replicates per parameter set to increase with the number of spatial units being studied. Ultimately, this mixed result points to a need to either formulate an algorithm that uses the bagged filter idea that is less computationally expensive or acknowledge this weakness and focus on a different method altogether. An algorithm that is gradient-based (instead of getting a precise likelihood estimate for each parameter set, we could get a noisy estimate that informs us about the direction in which to move our parameters) could yield an algorithm that is less demanding. Alternatively, one could follow one of the leads uncovered by the simulation studies: that the block particle filter is very competitive for all the problems on which we have tested it. An iterated block particle filter would require block-specific parameter sets since resampling is performed at the block level. Spatially correlated parameter perturbations that pull block parameters to the average of the parameters across blocks at the resampling steps can help ensure that parameter estimates in different blocks do not diverge.

The spatPomp package

I disucss the **spatPomp** package in Chapter 3. I demonstrate how one can perform a data analysis on a simple correlated Brownian motions example. I also show how to construct a coupled measles transmission SpatPOMP model, which forms the basis for a data analysis.

So far, one of the most useful purposes of this package is that a new method (e.g. an iterated block particle filter) can be implemented and tested in a matter of 1-2 weeks. There are already numerous example methods that have been implemented that each take advantage of the modularity of model components. The open-source nature of the package allows users to contribute new methods and the process of submission to the Comprehensive R Archive Network (CRAN) has ensured that the package is widely available across multiple platforms.

The package's main future work is support for multivariate observations for each spatial unit. The epidemiological applications on which we have used the package involve spatial units that each have a time series of case counts. However, multivariate observations for, say, multi-strain viruses like the dengue virus will become more commonplace and require some generalization of the available methods.

Overall, there are various open areas in which to contribute to this field. I anticipate new methods will keep being proposed until a standard method is found. I look forward to contributing to this field in any way that I can.

Bibliography

Ades, M. and Van Leeuwen, P. J. (2015). The equivalent-weights particle filter in a high-dimensional system. *Quarterly Journal of the Royal Meteorological Society*, 141(687):484–503.

Anderson, J., Hoar, T., Raeder, K., Liu, H., Collins, N., Torn, R., and Avellano, A. (2009). The data assimilation research testbed: A community facility. *Bulletin of the American Meteorological Society*, 90(9):1283–1296.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear, non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188.

Asfaw, K., Ionides, E. L., and King, A. A. (2021a). `spatPomp`: R package for statistical inference for spatiotemporal partially observed Markov processes. `https://github.com/kidusasfaw/spatPomp`.

Asfaw, K., Park, J., Ho, A., King, A. A., and Ionides, E. L. (2021b). Statistical inference for spatiotemporal partially observed Markov processes via the R package spatpomp. *arXiv:2101.01157*.

Bakker, K. M., Martinez-Bakker, M. E., Helm, B., and Stevenson, T. J. (2016). Digital epidemiology reveals global childhood disease seasonality and the effects of immunization. *Proceedings of the National Academy of Sciences*, 113(24):6689–6694.

Becker, A. D., Birger, R. B., Teillant, A., Gastanaduy, P. A., Wallace, G. S., and Grenfell, B. T. (2016). Estimating enhanced prevaccination measles transmission hotspots in the context of cross-scale dynamics. *Proceedings of the National Academy of Sciences*, 113(51):14595–14600.

Becker, A. D., Wesolowski, A., Bjørnstad, O. N., and Grenfell, B. T. (2019). Long-term dynamics of measles in London: Titrating the impact of wars, the 1918 pandemic, and vaccination. *PLoS Computational Biology*, 15(9):e1007305.

Becker, A. D., Zhou, S. H., Wesolowski, A., and Grenfell, B. T. (2020). Coexisting attractors in the context of cross-scale population dynamics: Measles in London as a case study. *Proceedings of the Royal Society of London, Series B*, 287(1925):20191510.

Bengtsson, T., Bickel, P., and Li, B. (2008). Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. In Speed, T. and Nolan, D., editors, *Probability and Statistics: Essays in Honor of David A. Freedman*, pages 316–334. Institute of Mathematical Statistics, Beachwood, OH.

Bentkus, V., Götze, F., and Tikhomoirov, A. (1997). Berry–Esseen bounds for statistics of weakly dependent samples. *Bernoulli*, 3(3):329–349.

Beskos, A., Crisan, D., Jasra, A., Kamatani, K., and Zhou, Y. (2017). A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability*, 49(1):24–48.

Bhadra, A., Ionides, E. L., Laneri, K., Pascual, M., Bouma, M., and Dhiman, R. C. (2011). Malaria in northwest India: Data analysis via partially observed stochastic differential equation models driven by Lévy noise. *Journal of the American Statistical Association*, 106:440–451.

Bjørnstad, O. N. and Grenfell, B. T. (2001). Noisy clockwork: Time series analysis of population fluctuations in animals. *Science*, 293:638–643.

Blackwood, J. C., Cummings, D. A. T., Broutin, H., Iamsirithaworn, S., and Rohani, P. (2013a). Deciphering the impacts of vaccination and immunity on pertussis epidemiology in Thailand. *Proceedings of the National Academy of Sciences of the USA*, 110:9595–9600.

Blackwood, J. C., Streicker, D. G., Altizer, S., and Rohani, P. (2013b). Resolving the roles of immunity, pathogenesis, and immigration for rabies persistence in vampire bats. *Proceedings of the National Academy of Sciences of the USA*.

Blake, I. M., Martin, R., Goel, A., Khetsuriani, N., Everts, J., Wolff, C., Wassilak, S., Aylward, R. B., and Grassly, N. C. (2014). The role of older children and adults in wild poliovirus transmission. *Proceedings of the National Academy of Sciences of the USA*, 111(29):10604–10609.

Brehmer, J., Louppe, G., Pavez, J., and Cranmer, K. (2020). Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences of the USA*, 117(10):5242–5249.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Bretó, C. (2014). On idiosyncratic stochasticity of financial leverage effects. *Statistics & Probability Letters*, 91:20–26.

Bretó, C. (2018). Modeling and inference for infectious disease dynamics: A likelihood-based approach. *Statistical Science*, 33(1):57–69.

Bretó, C., He, D., Ionides, E. L., and King, A. A. (2009). Time series analysis via mechanistic models. *Annals of Applied Statistics*, 3:319–348.

Bretó, C. and Ionides, E. L. (2011). Compound Markov counting processes and their applications to modeling infinitesimally over-dispersed systems. *Stochastic Processes and their Applications*, 121:2571–2591.

Brown, G. D., Porter, A. T., Oleson, J. J., and Hinman, J. A. (2018). Approximate Bayesian computation for spatial SEIR(S) epidemic models. *Spatial and Spatio-temporal Epidemiology*, 24:27–37.

Buhnerkempe, M. G., Prager, K. C., Strelioff, C. C., Greig, D. J., Laake, J. L., Melin, S. R., DeLong, R. L., Gulland, F., and Lloyd-Smith, J. O. (2017). Detecting signals of chronic shedding to explain pathogen persistence: Leptospira interrogans in California sea lions. *Journal of Animal Ecology*, 86(3):460–472.

Cappello, C., De Iaco, S., and Posa, D. (2020). covatest: An R package for selecting a class of space-time covariance functions. *Journal of Statistical Software*, 94(1):1–42.

Casella, G. and Berger, R. L. (1990). *Statistical Inference*. Wadsworth, Pacific Grove.

Chambers, J. M. (1998). *Programming with Data: A Guide to the S Language*. Springer Science & Business Media.

Chandler, R. E. (2013). Exploiting strength, discounting weakness: Combining information from multiple climate simulators. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1991):20120388.

Dalziel, B. D., Bjørnstad, O. N., van Panhuis, W. G., Burke, D. S., Metcalf, C. J. E., and Grenfell, B. T. (2016). Persistent chaos of measles epidemics in the prevaccination United States caused by a small change in seasonal transmission patterns. *PLoS Computational Biology*, 12(2):e1004655.

de Cellès, M. D., Magpantay, F. M., King, A. A., and Rohani, P. (2018). The impact of past vaccination coverage and immunity on pertussis resurgence. *Science Translational Medicine*, 10(434):eaaj1748.

Del Moral, P. and Guionnet, A. (2001). On the stability of interacting processes with applications to filtering and genetic algorithms. *Annales de l'Institut Henri Poincare (B) Probability and Statistics*, 37:155–194.

Del Moral, P., Moulines, E., Olsson, J., and Vergé, C. (2017). Convergence properties of weighted particle islands with application to the double bootstrap algorithm. *Stochastic Systems*, 6(2):367–419.

Del Moral, P. and Murray, L. M. (2015). Sequential Monte Carlo with highly informative observations. *Journal on Uncertainty Quantification*, 3:969–997.

Doucet, A., de Freitas, N., and Gordon, N. J. (2001). *Sequential Monte Carlo Methods in Practice.* Springer, New York.

Doucet, A. and Johansen, A. (2011). A tutorial on particle filtering and smoothing: Fifteen years later. In Crisan, D. and Rozovsky, B., editors, *Oxford Handbook of Nonlinear Filtering.* Oxford University Press.

Earn, D. J., He, D., Loeb, M. B., Fonseca, K., Lee, B. E., and Dushoff, J. (2012). Effects of school closure on incidence of pandemic influenza in Alberta, Canada. *Annals of Internal Medicine*, 156:173–181.

Ebert, E. E. (2001). Ability of a poor man's ensemble to predict the probability and distribution of precipitation. *Monthly Weather Review*, 129(10):2461–2480.

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162.

Evensen, G. and van Leeuwen, P. J. (1996). Assimilation of geostat altimeter data for the Agulhas Current using the ensemble Kalman filter with a quasigeostrophic model. *Monthly Weather Review*, 124:58–96.

Finkenstädt, B. F. and Grenfell, B. T. (2000). Time series modelling of childhood diseases: a dynamical systems approach. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 49(2):187–205.

Genolini, C. (2008). A (not so) short introduction to S4. Technical report, The R-Project for Statistical Computing.

Gneiting, T., Balabdaoui, F., and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 69:243–268.

Gordon, N., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings–F*, 140(2):107–113.

He, D., Dushoff, J., Day, T., Ma, J., and Earn, D. J. D. (2013). Inferring the causes of the three waves of the 1918 influenza pandemic in England and Wales. *Proceedings of the Royal Society of London, Series B*, 280:20131345.

He, D., Ionides, E. L., and King, A. A. (2010). Plug-and-play inference for disease dynamics: Measles in large and small towns as a case study. *Journal of the Royal Society Interface*, 7:271–283.

Ionides, E. L., Asfaw, K., Park, J., and King, A. A. (2021). Bagged filters for partially observed spatiotemporal systems. *arXiv:2002.05211v2*.

Ionides, E. L., Bhadra, A., Atchadé, Y., and King, A. A. (2011). Iterated filtering. *Annals of Statistics*, 39:1776–1802.

Ionides, E. L., Bretó, C., and King, A. A. (2006). Inference for nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the USA*, 103:18438–18443.

Ionides, E. L., Breto, C., Park, J., Smith, R. A., and King, A. A. (2017). Monte Carlo profile confidence intervals for dynamic systems. *Journal of the Royal Society Interface*, 14:1–10.

Ionides, E. L., Nguyen, D., Atchadé, Y., Stoev, S., and King, A. A. (2015). Inference for dynamic and latent variable models via iterated, perturbed Bayes maps. *Proceedings of the National Academy of Sciences of the USA*, 112(3):719—-724.

Jirak, M. (2016). Berry–Esseen theorems under weak dependence. *The Annals of Probability*, 44(3):2024–2063.

Johansen, A. M. and Doucet, A. (2008). A note on the auxiliary particle filter. *Statistics & Probability Letters*, 78:1498–1504.

Kain, M. P., Childs, M. L., Becker, A. D., and Mordecai, E. A. (2020). Chopping the tail: How preventing superspreading can help to maintain COVID-19 control. *MedRxiv*.

Kantas, N., Doucet, A., Singh, S. S., Maciejowski, J., Chopin, N., et al. (2015). On particle methods for parameter estimation in state-space models. *Statistical Science*, 30(3):328–351.

Katzfuss, M., Stroud, J. R., and Wikle, C. K. (2020). Ensemble Kalman methods for high-dimensional hierarchical dynamic space-time models. *Journal of the American Statistical Association*, 115(530):866–885.

Keeling, M. and Rohani, P. (2009). *Modeling Infectious Diseases in Humans and Animals.* Princeton University Press, Princeton, NJ.

Kevrekidis, I. G. and Samaey, G. (2009). Equation-free multiscale computation: Algorithms and applications. *Annual Review of Physical Chemistry*, 60:321–344.

King, A. A., Ionides, E. L., Pascual, M., and Bouma, M. J. (2008). Inapparent infections and cholera dynamics. *Nature*, 454:877–880.

King, A. A., Nguyen, D., and Ionides, E. L. (2016). Statistical inference for partially observed Markov processes via the R package pomp. *Journal of Statistical Software*, 69:1–43.

Lau, M. S., Becker, A. D., Korevaar, H. M., Caudron, Q., Shaw, D. J., Metcalf, C. J. E., Bjørnstad, O. N., and Grenfell, B. T. (2020). A competing-risks model explains hierarchical spatial coupling of measles epidemics en route to national elimination. *Nature Ecology & Evolution*, pages 1–6.

Lee, E. C., Chao, D. L., Lemaitre, J. C., Matrajt, L., Pasetto, D., Perez-Saez, J., Finger, F., Rinaldo, A., Sugimoto, J. D., Halloran, M. E., et al. (2020). Achieving coordinated national immunity and cholera elimination in Haiti through vaccination: a modelling study. *The Lancet Global Health*, 8(8):e1081–e1089.

Lei, J., Bickel, P., and Snyder, C. (2010). Comparison of ensemble Kalman filters under non-Gaussianity. *Monthly Weather Review*, 138(4):1293–1306.

Leutbecher, M. and Palmer, T. N. (2008). Ensemble forecasting. *Journal of Computational Physics*, 227(7):3515–3539.

Li, R., Pei, S., Chen, B., Song, Y., Zhang, T., Yang, W., and Shaman, J. (2020). Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2). *Science*, 368(6490):489–493.

Liu, J. S. (2001). *Monte Carlo Strategies in Scientific Computing.* Springer, New York.

Lorenz, E. N. (1996). Predictability: A problem partly solved. *Proceedings of the Seminar on Predictability*, 1:1–18.

Marino, J. A., Peacor, S. D., Bunnell, D., Vanderploeg, H. A., Pothoven, S. A., Elgin, A. K., Bence, J. R., Jiao, J., and Ionides, E. L. (2019). Evaluating consumptive and nonconsumptive predator effects on prey density using field time-series data. *Ecology*, 100(3):e02583.

Martinez-Bakker, M., King, A. A., and Rohani, P. (2015). Unraveling the transmission ecology of polio. *PLoS Biology*, 13(6):e1002172.

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models.* Chapman and Hall, London, 2nd edition.

Ng, B., Peshkin, L., and Pfeffer, A. (2002). Factored particles for scalable monitoring. *Proceedings of the 18th Conferenece on Uncertainty and Artificial Intelligence*, pages 370–377.

Ning, N., Ionides, E. L., and Ritov, Y. (2021). Scalable Monte Carlo inference and rescaled local asymptotic normality. *Bernoulli*, pre-published online.

Palmer, T. N. (2002). The economic value of ensemble forecasts as a tool for risk assessment: From days to decades. *Quarterly Journal of the Royal Meteorological Society*, 128(581):747–774.

Park, J. and Ionides, E. L. (2020). Inference on high-dimensional implicit dynamic models using a guided intermediate resampling filter. *Statistics & Computing*, 30:1497–1522.

Pitt, M. K. and Shepard, N. (1999). Filtering via simulation: Auxillary particle filters. *Journal of the American Statistical Association*, 94:590–599.

Pons-Salort, M. and Grassly, N. C. (2018). Serotype-specific immunity explains the incidence of diseases caused by human enteroviruses. *Science*, 361(6404):800–803.

Poterjoy, J. (2016). A localized particle filter for high-dimensional nonlinear systems. *Monthly Weather Review*, 144(1):59–76.

Ranjeva, S. L., Baskerville, E. B., Dukic, V., Villa, L. L., Lazcano-Ponce, E., Giuliano, A. R., Dwyer, G., and Cobey, S. (2017). Recurring infection with ecologically distinct HPV types can explain high prevalence and diversity. *Proceedings of the National Academy of Sciences*, page 201714712.

Rebeschini, P. and van Handel, R. (2015). Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866.

Roy, M., Bouma, M. J., Ionides, E. L., Dhiman, R. C., and Pascual, M. (2013). The potential elimination of Plasmodium vivax malaria by relapse treatment: Insights from a transmission model and surveillance data from NW India. *PLoS Neglected Tropical Diseases*, 7:e1979.

Shrestha, S., Foxman, B., Weinberger, D. M., Steiner, C., Viboud, C., and Rohani, P. (2013). Identifying the interaction between influenza and pneumococcal pneumonia using incidence data. *Science Translational Medicine*, 5:191ra84.

Shrestha, S., King, A. A., and Rohani, P. (2011). Statistical inference for multi-pathogen systems. *PLoS Computational Biology*, 7:e1002135.

Sigrist, F., Kunsch, H. R., and Stahel, W. A. (2015). spate: An r package for spatio-temporal modeling with a stochastic advection-diffusion process. *Journal of Statistical Software*, 63(14):1–23.

Snyder, C., Bengtsson, T., and Morzfeld, M. (2015). Performance bounds for particle filters using the optimal proposal. *Monthly Weather Review*, 143(11):4750–4761.

Tong, H. (1990). *Non-linear Time Series: A Dynamical System Approach.* Oxford Science Publ., Oxford.

van Kekem, D. L. and Sterk, A. E. (2018). Travelling waves and their bifurcations in the Lorenz-96 model. *Physica D: Nonlinear Phenomena*, 367:38–60.

Wallig, M. and Weston, S. (2019). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package.* R package version 1.0.15.

Wallig, M. and Weston, S. (2020). *foreach: Provides Foreach Looping Construct.* R package version 1.5.0.

Wesolowski, A., Eagle, N., Tatem, A. J., Smith, D. L., Noor, A. M., Snow, R. W., and Buckee, C. O. (2012). Quantifying the impact of human mobility on malaria. *Science*, 338(6104):267–270.

Wesolowski, A., Qureshi, T., Boni, M. F., Sundsøy, P. R., Johansson, M. A., Rasheed, S. B., Engø-Monsen, K., and Buckee, C. O. (2015). Impact of human mobility on the emergence of dengue epidemics in pakistan. *Proceedings of the National Academy of Sciences*, 112(38):11887–11892.

Wickham, H. (2019). *Advanced R.* CRC press.

Wickham, H., François, R., Henry, L., and Müller, K. (2020). *dplyr: A Grammar of Data Manipulation.* R package version 1.0.0.

Wikle, C. K., Zammit-Mangion, A., and Cressie, N. (2019). *Spatio-temporal Statistics with R.* CRC Press.

Xia, Y., Bjørnstad, O. N., and Grenfell, B. T. (2004). Measles metapopulation dynamics: A gravity model for epidemiological coupling and dynamics. *American Naturalist*, 164(2):267–281.