Simulation based inference in epidemic models

Master Thesis in Bio-statistics (STA495)

submitted in partial fulfillment of the requirements for the master degree in

Bio-statistics of Zurich University

by

Gilles Kratzer

03-809-142

supervised by

Associate Professor Michael Höhle, Stockholm Universitet

Full Professor Leonhard Held, Universität von Zürich

Stockholm, October 2015

Au printemps le sommeil ne cesse dès l'aurore
Partout se font ouïr les gazouillis d'oiseaux
La nuit s'achève enfin dans le souffle des eaux,
Qui sait combien de fleurs seront tombées encore ?

Mong-Kao-Jen

Abstract

The mathematical modelling of infectious diseases is a tool to study the mechanisms by which diseases spread, to predict the future course of an outbreak and to evaluate strategies to control an epidemic. In the class of compartmental models, which serve as a base mathematical framework for understanding the complex dynamics of infectious diseases, the Susceptible-Infectious-Recovered (SIR) model is a valuable model for many infectious diseases.

Typically, stochastic model's randomness vanishes for a large population size. This is an issue for modelling infectious diseases that exhibits high seasonal stochasticity. In the stochastic SIR modelling context, infectious diseases dynamic is described by a Partially Observed Markov Process (POMP). A framework to perform data driven simulation based inference over POMP objects is presented with a focus on stochastic epidemic models. The class of simulation model that is presented is an Euler multinomial with extra stochasticity added to the transition rates. The inference algorithm, for this class of model, is based on a Sequential Monte Carlo algorithm for hidden states inference, and the parameter inference is based on Maximum Likelihood Estimation via Iterated Filtering algorithm as proposed by Breto et al. (2009).

As an example, a single age strata without seasonality with extra noise is used to fit routine collected public health surveillance data about rotavirus data in Germany. Computational investigations are presented to show that, based on those preliminary results, this framework is suitable to infer a more complicated model from the data.

Acknowledgements

I would like to thank Prof. Leonhard Held who helped me find an internship abroad and agreed to be my official supervisor at Zurich University. I am especially grateful to Dr. Michael Höhle who hosted me at Stockholm University. He gave me valuable comments all along my Master Thesis and was very patient and enthusiastic. I would also like to thank Theresa Stocks for the motivating and stimulating discussions. A special thanks goes to Andreas Hicketier for the lunches and German discussions we had, for showing me the many wonders of Stockholm and most of all for sharing with me the largest office of Stockholm University. Thank to Dr. Eva Furrer who gave me the necessary support to build this project and more generally for her help during all my studies at Zurich University.

Finally, I am more than grateful to Catarina Da Costa for the unconditional support she offered me during these months. Without her, none of this would have been possible.

Contents

1 Introduction

The mathematical modelling of infectious diseases is a tool which has been used for decades to study the mechanisms by which diseases spread, to predict the future course of an outbreak, to evaluate strategies to control an epidemic and to help create public health interventions [Daley, Gani, and Gani, 2001]. Mathematical modelling of infectious diseases has an outstanding history of success from pioneer work of Bernoulli which successfully defended the general inoculation against smallpox to enhance life expectancy [Graunt, 1939], to recent work [Brauer, Castillo-Chavez, and Castillo-Chavez, 2001] where mathematical models of epidemics continually asses strategies to control them.

Weidemann, Dehnert, Koch, Wichmann, and Höhle [2014a] evaluate the impact of vaccination against rotavirus in Germany concluding that a systematic vaccination leads to mortality reduction. One limitation of the approach used in this study is that modelling is deterministic and extra stochasticity has to be added in order to take into account seasonal variations. The purpose of this Master Thesis is to describe and perform preliminary tests of a framework that is able to carry out simulation based inference on rotavirus data using a stochastic modelling approach. Hopefully leading to an improvement of the model's predictive or evaluation ability.

1.1 Mathematical modelling of outbreaks

A well established method to operate inference on epidemic models, is to divide the population of interest by the abstract notion of compartments, defined by health status with respect to the pathogen in the system and demographic or epidemiological features. One of the cornerstone works of such compartmental models was done by Kermack WO [1927]. In this paper, a deterministic compartmental model was successfully described and applied to real data. Following this work, the class of compartmental models, called Susceptible-Infectious-Recovered (SIR) model was extendly use. A large variety of extension exists. In compartmental models, individuals are assigned to different compartments, each representing a specific stage of the epidemic or a demographic status.

The SIR model [Kermack WO, 1927] is a model that assumes a fixed population size and only three compartments. $S(t)$ is the number of individuals not yet infected with the disease

at time t, thus susceptibles to the disease. $I(t)$ is the number of individuals who have been infected with the disease at time t and spread the disease to those in the susceptible category. $R(t)$ is the compartment used for those individuals who have been infected and then recovered from the disease at time t. Figure 1.1 shows a schematic representation of an SIR model where black arrows between compartments represents the flux of individuals. Each flux of individuals is associated to a rate.

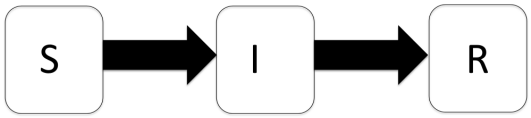

Figure 1.1: Schematic representation of an SIR model

Compartmental models differ in whether they consider the infection process as a deterministic or, as a stochastic process. The following sections introduce both model formulations and the issue of stochastic modelling's randomness vanishment.

1.1.1 Deterministic modelling

When dealing with large populations deterministic models are often used. In deterministic compartmental models, the transition rates from one compartment to another are mathematically expressed as derivatives. Hence the model is formulated using ordinary differential equations (ODE), Kermack WO [1927]. A closed population is assumed i.e at all times t: $P(t) = S(t) + I(t) + R(t)$.

- $\frac{dS(t)}{dt} = -\beta S(t)\frac{I(t)}{P(t)}$
- $\frac{dI(t)}{dt} = \beta S(t)\frac{I(t)}{P(t)} - \gamma I(t)$
- $\frac{dR(t)}{dt} = \gamma I(t)$

Several assumptions were made in the formulation of the above equations. The population in a compartment is differentiable with respect to time and the epidemic process is deterministic. The compartments are homogeneous, meaning that an individual in the population contracts the disease with a rate of β, being the *infection rate* of the disease. A fraction equal to γ representing the mean recovery/death rate of infectives who are leaving this compartment per unit time to enter the recovered compartment. Here $1/\gamma$ is called the mean infective period. These processes occurring simultaneously, are referred to as the *Law of Mass Action*, implying that the rate of contact between two groups in a population is proportional to the size of each of the groups concerned. Finally, it is assumed that birth and death process in the population are much slower than the rate of infection and recovery, thus can be ignored.

In deterministic settings, the *Basic reproduction number* R_0 is given by:

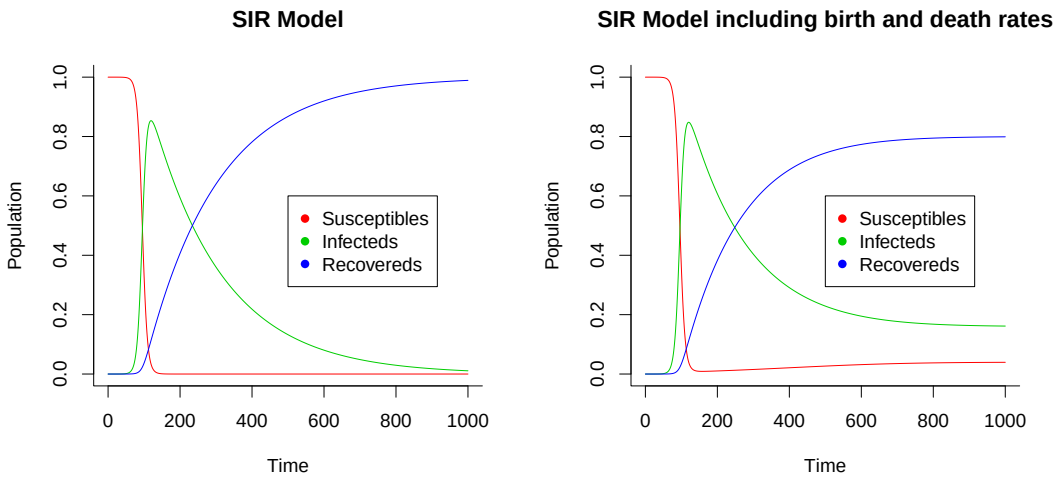

Figure 1.2: Deterministic SIR model

$$R_0 = \frac{\beta}{\gamma} N$$

This is the number of new infections produced by one infected individual in a intact population consisting only of susceptibles. When $R_0 > 1$, the epidemic takes off and when $R_0 < 1$, there is no epidemic [Diekmann, Heesterbeek, and Metz, 1990].

The deterministic SIR model can be adapted to a more sophisticated setting including birth and death processes for example, as shown in figure 1.2 Initial distribution of population is $S(0) = 1 - 10^{-6}$, $I(0) = 10^{-6}$ and $R(0) = 0$. Model parameters are $\beta = 0.15$ and $\gamma = 0.005$. As one can see, in adding a birth/death rate $\mu = 0.001$ the dynamic changes completely. Indeed without birth and death rates the epidemic extinct quickly and with birth and death rates the dynamic change drastically. The adaptability and the fast and easy inference process made the deterministic compartmental model a very popular model for modelling outbreaks. Solvers exist and are based on very efficient algorithm such as Runge-Kutta or Euler scheme.

Deterministic SIR models are often used in the case of sufficiently large populations, i.e in the thermodynamic limit. Because the larger the population is the better the assumption of homogeneity (i.e each individual in a given class is equivalent to the others) is met [Bartlett, 1957]. They can be viewed as limit models of a stochastic model class.

1.1.2 Stochastic modelling

A stochastic SIR model is defined analogously as the deterministic model. A closed homogeneous population is assumed, and $S(t)$, $I(t)$ and $R(t)$ have the same definition as in the deterministic setting. As done previously, birth and death are ignored in this simple setting. The dynamic of the model is defined as follows. Infectives have contact with susceptibles at a constant rate β. Contacts are mutually independent. Any susceptible which is in contact with

an infected individual immediately becomes infective and starts spreading the disease following the same rules. Infected individuals remain infectious for a random amount of time governed by an infections period distribution after which they stop being infectious and recover. The infectious periods are defined to be independent and identically distributed (also independent of the contact processes). The exponential distribution with intensity parameter γ has received special attention in the literature, because the resulting model is Markovian [Britton and Giardina, 2014]. A stochastic process is said to be a Markov process or have the markov property if the conditional probability of the future state conditional on both present and past states depends only on the present state of the process.
In continuous time setting, Table 1.1.2 specifies a Markovian stochastic SIR model in giving the set of events and rates.

Event	Rate
$(S(t), I(t)) \rightarrow (S(t)-1, I(t)+1)$	$\beta S(t)\frac{I(t)}{P(t)}$
$(S(t), I(t)) \rightarrow ((S(t), I(t)-1))$	$\gamma I(t)$

Table 1.1: Specification of a stochastic SIR model

where $\beta \in \mathbb{R}^+$ and $\gamma \in \mathbb{R}^+$ and recovery time exponentially distributed. $R(t)$ is implicitly given as the total number of individual is fixed.

A simple and efficient algorithm to simulate a stochastic model is the *Gillespie algorithm* [Gillespie, 1977]. This algorithm assumes that all possible transitions between compartments occur independently and are simulated at each time step with constant probability per unit time that depends on the current state of the system (i.e the number of individual in each compartment). The idea is to sample the next time event from an exponential distribution. Then, the event is randomly chosen amongst the possible transitions between compartments with probabilities proportional to their individual rates. Figure 1.3 shows two realizations of an stochastic SIR model simulated by a Gillespie algorithm (see annex 7.1 for the R code), for two different population size. Model parameters are : $\beta = 0.02$, $\gamma = 0.3$ and $\mu = 0.1$

The *basic reproduction number* in the stochastic setting is the average number of secondary cases directly caused by an infectious case in an entirely susceptible population. For the stochastic SIR model, as defined in table 1.1.2, R_0 can be calculated as [Greenwood and Gordillo, 2009]:

$$R_0 = \frac{\beta}{\gamma} S(0)$$

1.1.3 Stochasticity vanishment

A major issue of the stochastic modelling approach is that for large population size the stochasticity decreases. In the infinite limit the stochasticity vanishes [Kurtz, 1970]. Thus, stochastic

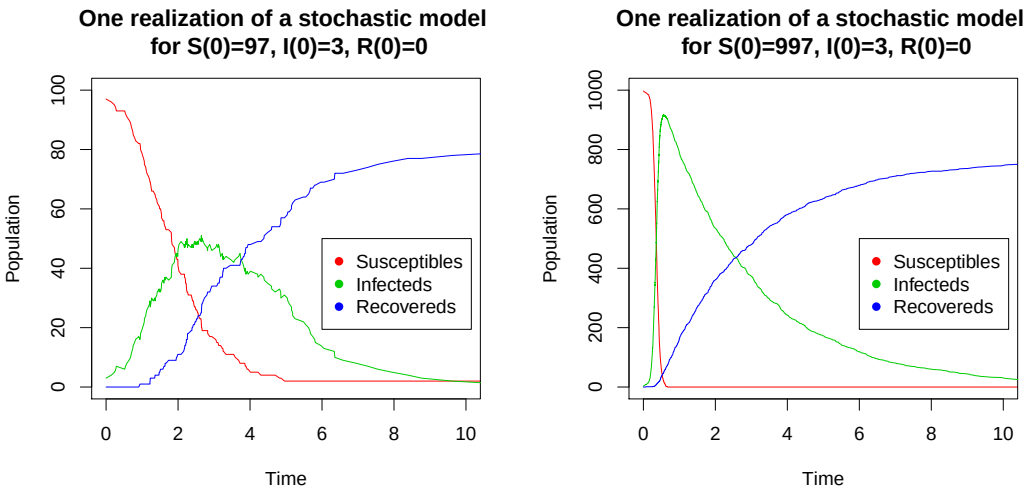

Figure 1.3: Realizations of stochastic models for different population size

models (specified in table 1.1.2) fail to appropriately reproduce the stochastic dynamics, as the stochasticity depends on the sample size. Indeed, there is a decrease of the relative stochasticity of trajectories when several realizations of a stochastic models are shown. Figure 1.4 (see annex 7.1 for the R code) shows the decline in stochasticity using 100 realizations of the number of infected for small (100 individuals) and large populations (1000 individuals) for a given set of parameters : $\beta = 0.02$, $\gamma = 0.3$ and $\mu = 0.1$.

There is a need to investigate other approaches that include stochastic noises that do not vanish for large sized population. The idea, which will be developed in the next chapters, is to add white noise to the transition rates in a stochastic SIR model. The advantages of white noises are that it depends on a unique parameter (only the intensity is modeled not the frequency), that the Markov property is inherited by the dynamical system and finally white noise can be designed to model colored noise.

This thesis is organized as follows, chapter 2 introduces the rotavirus data which will be use to exemplify the simulation based inference method. Chapter 3 contains the theoretical background of the thesis, it introduces the simulation model that allows scalable stochasticity also the parameter inference methods to investigate models based on real data. Chapter 4 describes the implementation of the method using an existing R package. Chapter 5 presents the results obtained in applying this inference method to rotavirus data.

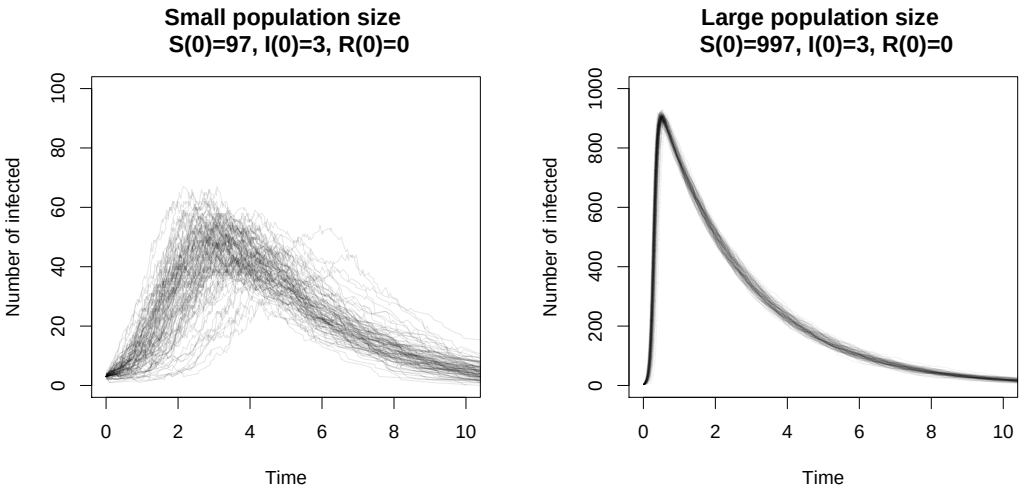

Figure 1.4: 100 trajectories simulations of Stochastic models for different population sizes

2 Description of the epidemiological data

This chapter presents the data used to exemplify the methods described in this research. First the epidemiological features of rotavirus transmission and infection are presented. Then the German reporting data are described. Finally, the estimates of basic reproduction number and the mean recovery time are given which will help to build and asses the model for rotavirus.

2.1 Rotavirus

Rotavirus (RV) infection is the most common cause of severe diarrheas and dehydration amongst infants and young children worldwide [Dennehy, 2000]. It is a genus of double-stranded RNA virus of the family Reoviridae which is endemic worldwide. Nearly every child in the world has been infected with RV at least once by the age of five. It is estimated that RV leads worldwide to annually more than 110 million episodes of diarrheas causing 25 million of clinical visits, 2 million of hospitalizations, and 453000 deaths [Tate, Burton, Boschi-Pinto, Steele, Duque, and Parashar, 2012]. Immunity to RV develops through infections, so subsequent infections are less severe. Adults are rarely affected. Five species of the virus exist, referred to A, B, C, D, and E. RV A, the most common species, causes more than 90% of RV infections in humans [Bernstein, 2009]. However, this thesis will not distinguish the different type of RV-infection.

Since 2001, acute rotavirus infection is notifiable in Germany [Koch and Wiese-Posselt, 2011]. Laboratory-confirmed cases are routinely reported to the local health offices and forwarded electronically via the state health authority to the Robert Koch Institute (RKI). The data-sets used in this thesis are based on the available surveillance reported data described in Weidemann, Dehnert, Koch, Wichmann, and Höhle [2014b] and which are available from github. It includes information on age and federal state of residency. The reported data were separated in two regions of eastern (EFS) and western (WFS) federal states of Germany (the state of Berlin is considered to be part of the western federal state). It is expected to have a difference of reported RV cases in between EFS and WFS due to difference in healthcare system [Rosner, Stark, and Werber, 2010].

In between 2001-2009, 503373 RV cases have been reported in Germany. 210712 have been reported in EFS and 292661 in WFS respectively. 144087 RV cases have been reported for chil-

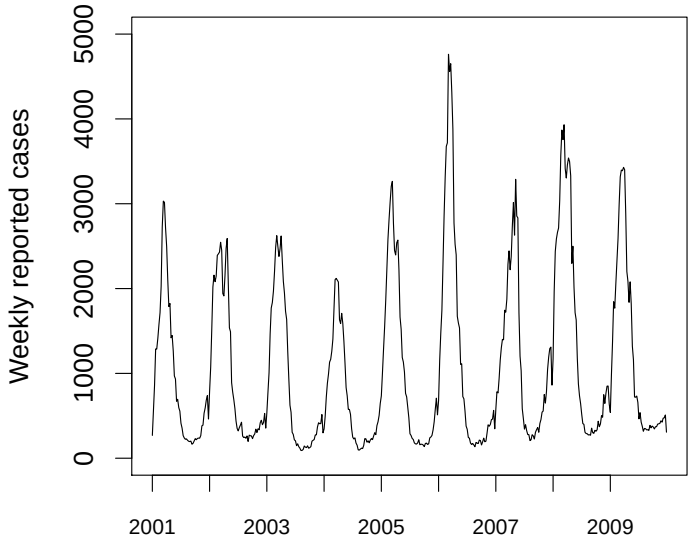

Figure 2.1: Unstratified weekly reported number of cases

dren < 5 years old in EFS and 210147 for WFS. For patients above 5 years, 66625 RV cases have been reported for EFS and 82514 for WFS. Additional demographic data on monthly birth rates, annual age-stratified mortality rates, age-stratified migration data from 1990-2009, and age-stratified population counts in the EFS and WFS for the years of 2001-2009 was obtained from the GENESIS database at the Federal Statistical Office of Statistics [2013]; Weidemann et al. [2014b].

Figure 2.1 presents the aggregated weekly number of reported number of RV. Figure 2.2 presents the weekly reported number of RV cases stratified by age (under and above 5 years old) and regions (EFS and WFS). As one can see, RV data exhibits a strong seasonal pattern with a maximum in March and minimum in August. Figure 2.3 shows the seasonality of the rotavirus aggregated data. We use, as a working definition of a rotavirus season, a cutoff between week 37 and 38 of the year, thus in September. The red line is the data for 2002 (from September 2002 to September 2003), and each other line is another year of data. The rotavirus season 2002 will be used for model fitting This is the first complete season, as the data for season 2001 starts in January. It can be observed that there is a high stochasticity in between years. The original data are available in both EFS WFS and for 10 age strata (0, 1, 2, 3, 4, 5-19, 20-30, 40-59, 60-79, 80+).

As previously pointed out, Weidemann et al. [2014b] assess with a deterministic model the effect of rotavirus vaccination in using the difference of healthcare system in between West and East side of Germany. The ODE system describes the dynamic of the infection while an observational model based on Negative binomial distribution is used (NB). However, one limitation

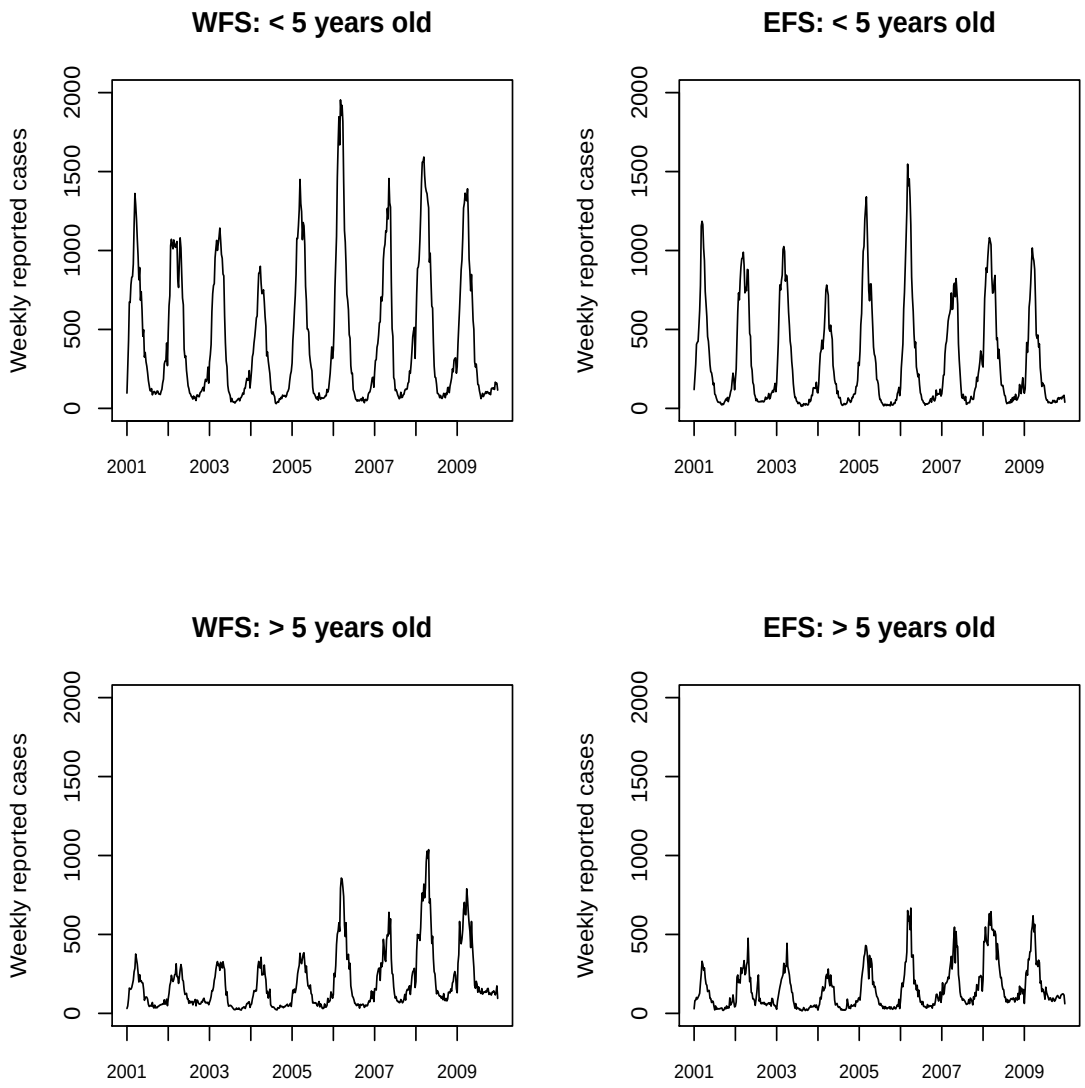

Figure 2.2: Weekly reported number of cases stratified by age and region

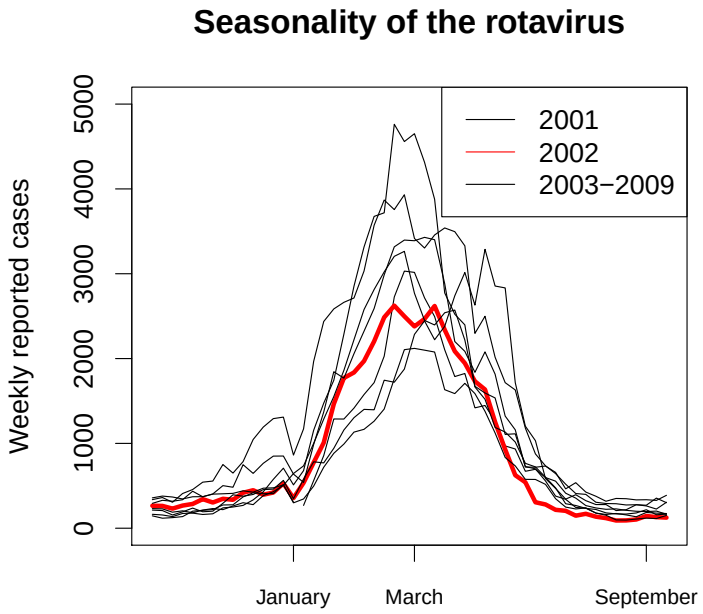

Figure 2.3: Seasonality of the weekly reported number of cases

of this approach is that the stochasticity arises from the observational model only. A stochastic approach relying on stochastic modelling of the dynamic of the rotavirus infection process, is believed to better integrate seasonal stochasticity (see figure 2.3). This is the main motivation to use a stochastic setting to model the rotavirus dynamic.

The incubation period for the rotavirus diarrhea is usually less than 48 hours [Atkinson, Wolfe, and Hamborsky, 2011]. The transmission of the rotavirus is by fecal-oral and thus involve close person-to-person contact or fomites. Those are objects or materials that can carry infections, such as clothes, utensils, furniture, toys or contaminated stools. Rotavirus is highly communicable.

The basic reproduction number is estimated to be as high as $R_0 = 60$. This estimate was obtained by taking the mean of the basic reproduction number for 15 country reproduced in Pitzer, Viboud, Lopman, Patel, Parashar, and Grenfell [2011]. The data used are reproduced in 7.1. The mean recovery time can be estimated as one week. "The gastrointestinal symptoms generally resolve in 3 to 7 days" [Committee et al., 1996]. Then the mean recovery time can be estimated as $\gamma = 1$ in a weekly period of time. These estimates will be useful for modelling and for model assessment of rotavirus.

3 Theoretical background

This chapter introduces a simulation based inference method. Firstly, the problem we tackle in this thesis will be presented. Then an introduction to the mathematical formulation of the Continuous Time Markov Chains is given. This is the framework in which the Euler multinomial algorithm is described. The last two sections are devoted to the inference algorithms: to infer the internal states we used the Sequential Monte Carlo algorithm and to infer the model parameters we used a likelihood based algorithm.

3.1 Introduction to Inference algorithms for Partially Observed Markov Process : Problem statement

The purpose of this thesis is to perform simulation based inference for the model parameters on an epidemic model that allows tuneable stochasticity. To do so, the RV incidence data will be modeled as a Partially Observed Markov Process (POMP) also called Hidden Markov Model (HMM) or state-space model. A POMP is a statistical Markov Model in which the hidden states are assumed to have the Markov property and emit symbols or observations. In the particular case of SIR model, the hidden states are the number of individuals in the compartments. A POMP can be represented as a graphical network as shown in figure 3.1. This graphical model is a set of nodes and arrows. The nodes are the states and the arrows encode the conditional probabilities. In figure 3.1, the Markov process generating the hidden states is depicted in black and the emission process in blue. In the specific context of SIR modelling, hidden states are the status of the SIR compartments and the observations process is the number of newly reported cases.

The reason why a POMP is a good candidate for modelling infectious disease is that the number of individuals susceptible, infected and recovered are unknown. The time series of the newly reported cases (which is basically the only available information) is a noised signal of the total number of newly infected people in the population. To model the RV data we use the a Markov process as described in table 1.1.2 and the observational model is a negative binomial trial of the number of newly infected. The rational to use a negative binomial distribution is that this distribution allows to deal with over dispersion. The total number of infected at time

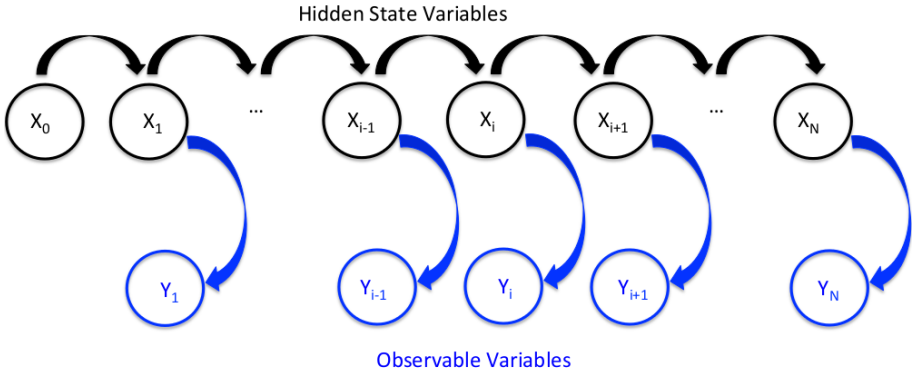

Figure 3.1: Schematic representation of an Hidden Markov Model (HMM)

t is given by: $I(t)$, the number of newly infected at time t is given by: $i(t) = \beta I(t)S(t)$. The newly reported number of infected is then modeled by $H(t) \sim \text{NB}(\text{size} = \theta\ i(t), \text{prob} = \rho)$. The negative binomial distribution is parametrize by two parameters: θ is called the shape parameter and ρ is called the probability. Alternative parametrization exists. This model is a time point wise model, thus in discrete time.

In order to give an example of the observational model, figure 3.2 depicts one simulation of the total number of infected and the number of newly infected. Then the number of newly reported cases is given by a point wise negative binomial trial with $\theta = 0.05$ and $\rho = 0.65$.

A well suited inference framework for POMP is the Sequential Monte Carlo (SMC) algorithm also called Particle Filter algorithm described by Doucet, De Freitas, and Gordon [2001] and Arulampalam, Maskell, Gordon, and Clapp [2002]. The purpose of the SMC algorithm is to sequentially estimate the density of hidden states given the observations. The system is observed at times $t \in \{1, \cdots, N\}$ with values $\{y_1, \cdots, y_N\}$. The density of hidden states is represented by a set of particles. Each particle has a weight which represents the local mass of the density function. As the estimations of the density are done sequentially, no other re-computations are necessary to estimate the updated density other than incorporating new observation information. Additional features such as re sampling step and kernel density estimation are specifically designed to deal with degeneracy problem and sample impoverishment problem respectively, which are two major issues of the SMC algorithm.

Parameter estimation for POMP is classically done by Expectation-Maximization algorithm (i.e Baum-Welch algorithm) [Bilmes et al., 1998]. But due to the complexity of the likelihood, evaluation could be rather complicated [Britton and Giardina, 2014]. For a more detailed discussion refer to section 4.1. Ionides, Bretó, and King [2006] develops and implements a parameter inference strategy for POMP that assumes that the parameter is a time varying random process (i.e a random walk), because inference is easier in time varying setting, and takes the limit of the estimate for null variance.

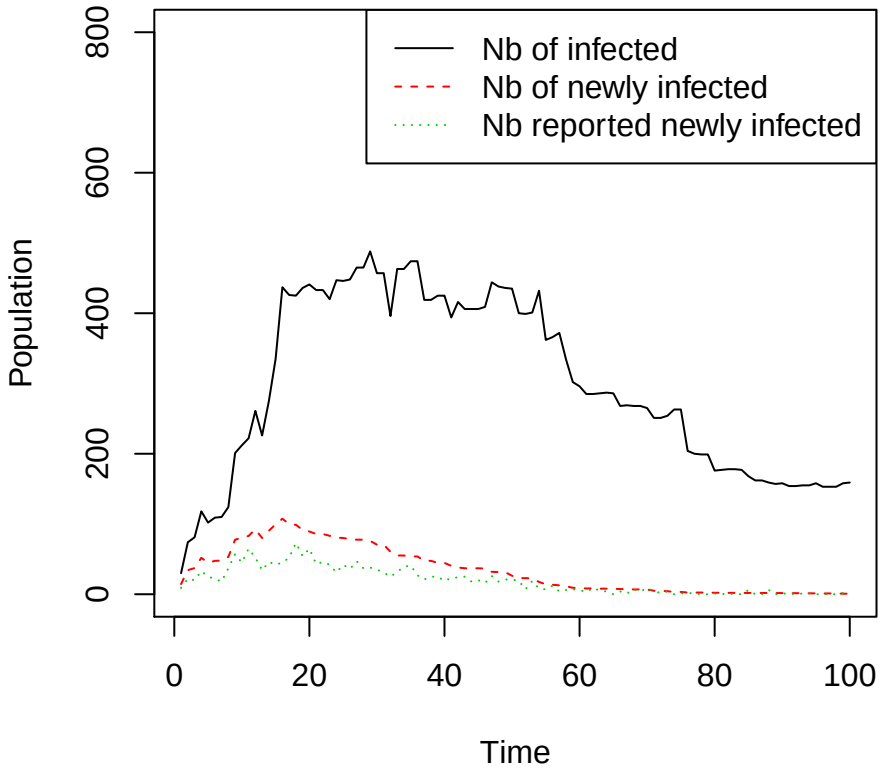

Figure 3.2: Representation of the weekly number of reported cases, the weekly new number of cases and a negative binomial noised signal of the new weekly number of reported cases

3.2 Continuous Time Markov Chain

The aim of this section is to present the necessary mathematical background to understand Continuous Time Markov Chain. In annex 7.3 two useful concepts are introduced: filtration process and the Poisson Random Measure (PRM).

To define a Continuous Time Markov Chain (CTMC) [Anderson, 2012], let there be a time dependent state vector $X_t = X(t;\omega) \in \mathbb{Z}_+^{N_c}$ with $\omega \in \Omega$ and where N_c is the number of compartments of the model. For an homogeneous SIR model we would have $N_c = 3$ (susceptible, infected and recovered). This state vector counts at time t the number of individuals in each of the N_c compartments. In a discrete state framework, the state vector process is right continuous with respect to time. This is notationally stressed by $X(t-)$. In order to describe the transitions between compartments, let us have a rate function $r : \mathbb{Z}_+^{N_c} \to \mathbb{R}_+$ and the stoichiometric coefficients matrix $s \in \mathbb{Z}^{N_c}$. Every flow of individuals between compartments is associated with a rate. The stoichiometric coefficients specify intensity of connections in between compartments. Then a CTMC is defined as:

$$dX_t = s\mu(dt) = s\mu(r(X(t-)); dt)$$

where μ is a scalar Poisson Random Measure [Çınlar, 2011] (See annex 7.3 for more details). The expectation of the Poisson Random Measure is given by $\mathbf{E}[\mu(dt)] = r(X(t-))dt$ with exponentially distributed waiting time. The generalization to a non-scalar counting measures is straight forward. The vector intensity of N_t transitions is defined by $R : \mathbb{Z}_+^{N_c} \to \mathbb{R}_+^{N_t}$ and a stoichiometric matrix $\mathbb{S} \in \mathbb{Z}^{N_c \times N_t}$. Then a general CTMC is given by:

$$d\boldsymbol{X}_t = \mathbb{S}\boldsymbol{\mu}(dt) = \mathbb{S}\left[\mu_1(dt), ..., \mu_{N_t}(dt)\right]^\top, \text{ for all } k, \mu_k(dt) = \mu(R_k(X(t-)); dt) \tag{3.1}$$

As an example, let us take the stochastic SIR model, defined in table 1.1.2. The state vector is given by $X(t) = [x_1(t), x_2(t), x_3(t)]$, where $x_i(t)$ for $i \in \{1, 2, 3\}$ has to be understood as the number of individuals in the susceptible, infected and recovered compartments in function of time. The stochiometric coefficient defined by table 1.1.2 is:

$$\mathbb{S} = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$$

The vector intensity is given by $R(X(t)) = [\beta x_1(t)x_2(t), \gamma x_3(t)]^\top$. Then the CTMC of this model is:

$$d\boldsymbol{X}_t = d\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = \mathbb{S}\boldsymbol{\mu}(dt) = \begin{bmatrix} -\beta x_1(t)x_2(t) & 0 \\ \beta x_1(t)x_2(t) & -\gamma x_3(t) \\ 0 & \gamma x_3(t) \end{bmatrix}$$

3.3 Construction of the Euler Multinomial class model

The stochasticity of a dynamical model is defined as the relative stochasticity in between different realizations of simulated trajectories. The stochasticity is said to be tunable or scalable if there exists a parameter that determines the intensity of the stochasticity. An *implicit* model is a statistical model for a stochastic system which is specified by a simulation algorithm. Parameter inference methods on *implicit* models are said to have the *plug-and-play* property (for more details see 4.1). The following developments follow the description given by [Bretó, He, Ionides, and King, 2009]. The general idea is to develop an algorithm for time series analysis based on compartmental models that include both a scalable stochasticity and the plug-and-play property. The scalable stochasticity will be achieved by adding white noise to the rates in between compartments. Another distinction compared to the previous section is that the model we want to construct should be in a discrete time setting. This will be achieved by constructing a CTMC as a limit of a discrete-time multinomial processes Cai and Xu [2007].

The characteristic of the compartmental models is that they respect total conservation of the mass balance. This implies that the total number of individuals of the system stays constant.

This can be seen as flows in between compartments. Let us introduce the number of count at time t between compartment i to compartment j written as $N_{ij}(t)$. Each number of count is associated with a rate $\mu_{ij} = \mu_{ij}(t, X(t))$ between compartment i to compartment j. A number of count is a random measure. For sake of simplicity, we will assume in the rest of the development that the matrix $N_{ij}(t)$ is a square matrix (with possibly zero rows and/or zero columns). Then equation 3.1 can be rewritten as:

$$x_i(t) = x_i(0) + \sum_{j \neq i} N_{ij}(t) - \sum_{j \neq i} N_{ji}(t).$$

It is very natural to assume that $x_i(t) \geq 0 \; \forall i \in \mathbb{N}$ and $\forall t \in \mathbb{R}$. This implies that the total population does not change over time. As pointed out previously, white noise (WN) has to be added to the transition rates in order to create the desired extra stochasticity. The WN should have the following properties in order to respect the closeness of the population:

1. Independent increments: The set of temporal increments of the WN should be such that: $\{\Gamma_{i,j}(t_2) - \Gamma_{i,j}(t_1), 1 \leq i \leq N_c, 1 \leq j \leq N_c\}$ is presumed to be independent of $\{\Gamma_{i,j}(t_4) - \Gamma_{i,j}(t_3), \text{ for all } i, j \in \{1, \cdots, N_c\}$ and all sets of ordered time points $t_1, t_2, t_3, t_4\}$
2. Stationary increments: The set of temporal increments of the WN should be such that: $\{\Gamma_{i,j}(t_2) - \Gamma_{i,j}(t_1), \text{ for all } i, j \in \{1, \cdots, N_c\}\}$ has a joint distribution depending only on $t_2 - t_1$
3. Non-negative temporal increments of the WN: $\{\Gamma_{i,j}(t_2) - \Gamma_{i,j}(t_1), \text{ for all } i, j \in \{1, \cdots, N_c\} \; \forall \, t_1 < t_2$
4. Unbiased multiplicative noise :$\mathbf{E}[\Gamma_{i,j}(t)] = t$
5. Partially independent noises: For each i, $\{\Gamma_{i,j}(t)\}$ is independent of: $\{\Gamma_{i,k}(t)\} \; \forall \; j \neq k$
6. Independent noises: $\{\Gamma_{i,j}(t)\}$ is independent of $\{\Gamma_{k,l}(t)\}$ for all $(i, j) \neq (k, l)$
7. Gamma noises marginally distributed $\Gamma_{i,j}(t + \delta) - \Gamma_{i,j}(t) \sim \text{Gamma}(\delta/\sigma^2_{i,j}, \sigma^2_{i,j})$

Condition number 7, introduces a certain distribution for the WN. The choice of the gamma noise is a convenient choice, because it fulfills the above requirements. In such a parametrization, the shape parameter is $\delta/\sigma^2_{i,j}$ and the scale parameter is $\sigma^2_{i,j}$. Thus the mean is given by δ and the variance is given by $\delta\sigma^2_{i,j}$.

The model is specified by the following equations. Let us define the number of compartment transitions by $\Delta N_{ij} := N_{ij}(t + \delta) - N_{ij}(t)$ and the gamma noises increments by $\Delta\Gamma_{ij} := \Gamma_{ij}(t + \delta) - \Gamma_{ij}(t)$, where i and j are compartments. The increment of the transition of individuals between compartments and the WN increment are both independent of time. Because they are dependent on the parameter δ. Then following Bretó et al. [2009]:

$$P(\Delta N_{ij} = n_{ij},\ \forall\, 1 \leq i \leq N_c, 1 \leq j \leq N_c \text{ and } i \neq j \mid X(t))$$
$$= \mathbf{E}\left[\prod_{i=1}^{N_c}\left\{\binom{x_i}{n_{i1},\cdots n_{ii-1}n_{ii+1}\cdots n_{iN_c}r_i}\left(1-\sum_{k\neq i}^{N_c} p_{ik}\right)^{r_i}\prod_{j\neq i} p_{ij}^{n_{ij}}\right\}\right] + o(\delta) \quad (3.2)$$

where $X(t) = \{x_i, i \in 1 : N_c\}$ is the set of the states of the compartments at time t and $r_i = x_i - \sum_{i\neq k} n_{i,k}$. Furthermore:

$$\begin{aligned} p_{ij} &= p_{ij}(\{\mu_{i,j}(n\delta, X(n\delta))\}, \{\Delta\Gamma_{i,j})\}) \\ &= (1 - \exp(-\sum_k \mu_{i,k}\Delta\Gamma_{i,k}))\mu_{i,j}\Delta\Gamma_{i,j}/\sum_k \mu_{i,k}\Delta\Gamma_{i,k}. \end{aligned}$$

Bretó et al. [2009] proves that 3.2 defines a CTMC when conditions 1-5 hold and that if conditions 1-7 hold, then infinitesimal transitions probabilities can be computed exactly. The exact computation is done using a Gillespie algorithm but in practice an Euler scheme is a reasonable approximation. This CTMC is *implicit* since numerical solutions are available at arbitrary precision. Indeed, exact computation can be done by an Gillespie algorithm and an arbitrary precision can be achieved by considering smaller δ (as shown in 3.2). Finally, the rationale behind introducing white noise and not colored noise is that it requires fewer parameters (only intensity) and that colored noise can be obtain by coupling white noises. Algorithm 1 shows the implementation of CTMC presented above.

Algorithm 1 Euler Multinomial model with gamma noise

1: Divide the interval [0,T] into N intervals of width δ =T/N
2: Set initial value X(0)=[S(0),I(0),R(0)]$^\top$
3: **for** $n = 0$ to $N - 1$ **do**
4: Generate noise increments
5: $\{\Delta\Gamma_{ij} = \Gamma_{ij}(n\delta + \delta) - \Gamma_{ij}(n\delta) \sim \text{Gamma}(\delta/\sigma_{i,j}^2, \sigma_{i,j}^2)\}$
6: Generate process increments
7: $(\Delta N_{i,1}, \cdots, \Delta N_{i,i-1}, \Delta N_{i,i+1}, \Delta N_{i,c}, R_i)$
8: $\sim \text{Multinomial}(X_i(n\delta), p_{i,1}, \cdots, p_{i,i-1}, p_{i,i+1}, \cdots, p_{i,c}, 1 - \sum_{k\neq i} p_{i,k})$
9: where $p_{ij} = p_{ij}(\{\mu_{i,j}(n\delta, X(n\delta))\}, \{\Delta\Gamma_{i,j})\}) =$
10: $(1 - \exp(-\sum_k \mu_{i,k}\Delta\Gamma_{i,k}))\mu_{i,j}\Delta\Gamma_{i,j}/\sum_k \mu_{i,k}\Delta\Gamma_{i,k}$
11: **end for**

As an example, figure 3.3 (see annex 7.3 for the R code) shows 100 realizations of an stochastic SIR model (see table 1.1.2) simulated using algorithm 1 that lead to a conserved stochasticity even for a large population size. In this example the total population size has been chosen at 80,000,000 i.e the population size of Germany.

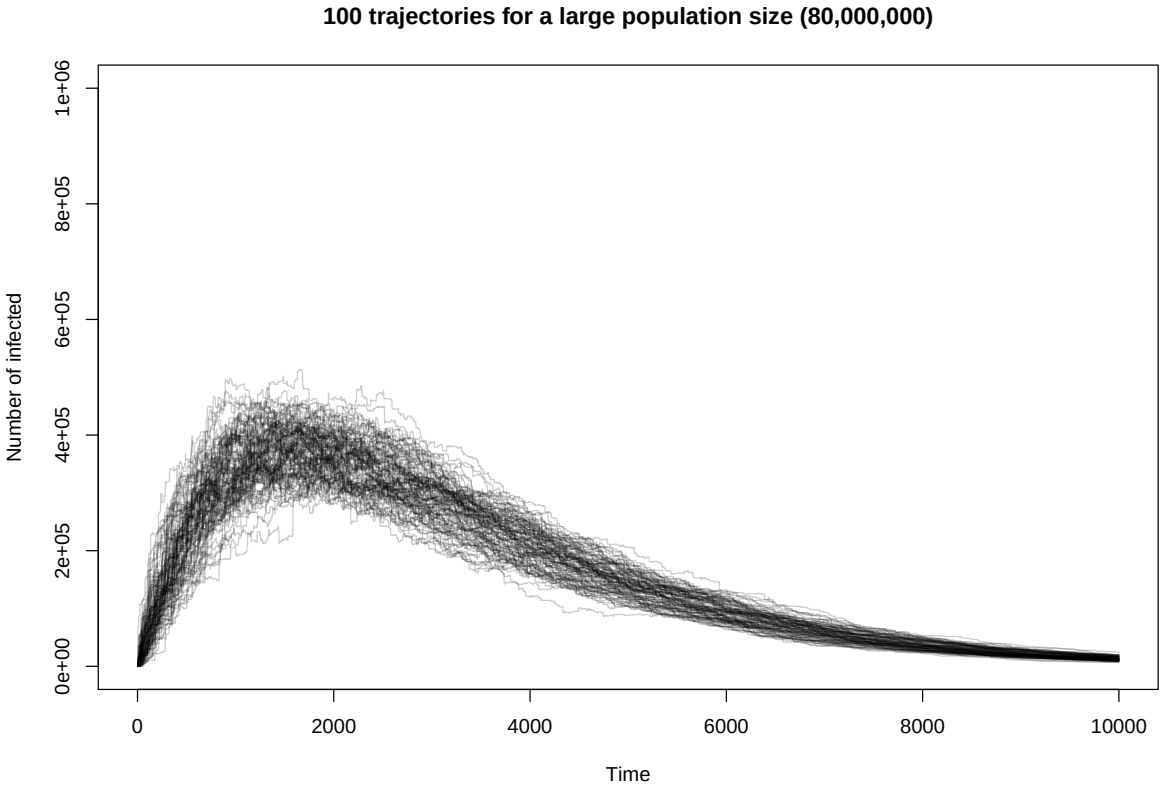

Figure 3.3: Stochasticity produced by Euler multinomial model

3.4 Monte Carlo method for POMP inference

The further computations follows Doucet et al. [2001] and Arulampalam et al. [2002]. The derivations are extended in order to become more digest and additional steps are added.

3.4.1 Naïve approach for POMP inference

For sake of simplicity we restrict our self to signals modeled as Markovian, nonlinear, non-Gaussian state-space models. The Markovian property seems very natural for infection contact driven problem i.e the number of infected at a certain time step depends exclusively on the number of infective at previous time step. A non-linear formulation comes naturally if the law of mass is admitted as transitions are proportional to product of the compartment size. The non-Gausiannity come from the fact that the considered transitions density in between hidden states is a noised gamma distributed process. This implies that the range of problem cover by this approach include problems solved by classical Kalman filter but also nonlinear and non-Gaussian problem.
Let us define a *Hidden Markov Model* (HMM) or a *Partial Observed Markov Process* (POMP). It is a statistical Markov Model in which the sequence of hidden states, $\{\mathbf{x}_t; t \in \mathbb{N}\}$, $\mathbf{x}_t \in \chi$ are assumed to have the Markov properties and result in observations $\{\mathbf{y}_t; t \in \mathbb{N}^*\}$, $\mathbf{y}_t \in \Upsilon$. Here, χ is the sample space of the signal (for example $\mathbb{N}^3$ for simple SIR model) and Υ is the sample space of the observations (for example $\mathbb{N}$ for reported cases in simple case). The initial distribution of the hidden states is given by $p(\mathbf{x}_0)$ and the transition equation is given by $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$. Observations are assumed to be conditionally independent given the process and of marginal distribution $p(\mathbf{y}_t \mid \mathbf{x}_t)$. Conditional independence imposes that $p(\mathbf{y}_t \mid \mathbf{x}_t, \mathbf{y}_{0:t-1}) = p(\mathbf{y}_t \mid \mathbf{x}_t)$. To summarize, an POMP is given by the following densities:

- $p(\mathbf{x}_0)$ (3.3)
- $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}), \forall t \geq 1$ (3.4)
- $p(\mathbf{y}_t \mid \mathbf{x}_t), \forall t \geq 1$ (3.5)

A POMP can be represented as a graphical model Cappé, Moulines, and Rydén [2006] as shown in figure 3.1. The time series of the signal is denoted by $\mathbf{x}_{0:t} = \{\mathbf{x}_0, \cdots, \mathbf{x}_t\}$ and the time series of the observations are denoted by $\mathbf{y}_{1:t} = \{\mathbf{y}_1, \cdots, \mathbf{y}_t\}$ up to time t. This notation is to stress that this is a set of vectors or scalar number. The aim is to estimate the *distribution* $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ or the *marginal distribution* $p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$ also called the *filtering distribution*. The filtering distribution is the probability that the system ends at time t in a certain state knowing all the observations. In addition, one can be interested to compute the expectation of a function of interest. This is of importance because any probability can be expressed as an expectation. For example, let us assume a probability space $(\Omega, \mathcal{F}, \mathbf{P})$ and let $X : \Omega \to \mathbb{R}$ be a random variable, $A \in \Omega$ an event and $\mathbf{I}$ the indicator function. Then one can express the probability that a realization of the random variable belongs to a set A as: $P(X \in A) = \mathbf{E}[\mathbf{I}(X \in A)]$. Thus for a function $f_t : \chi^{(t+1)} \to \mathbb{R}^{n_{f_t}}$ integrable with respect to $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ where $\chi^{(t+1)}$ is the sample space of the signal. Then the expectation of such a function is given by:

$$\mathbf{I}(f_t) = \mathbf{E}_{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}[f_t(\mathbf{x}_{0:t})] = \int f_t(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}$$

At any time $t \in \mathbb{N}_+$. Here, $\mathbf{I}(\cdot)$ is the expectation and $d\mathbf{x}_{0:t}$ is a measure over the sample space of the signal $\chi^{(t+1)}$. The distribution can be computed from *Bayes theorem*:

$$p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}) p(\mathbf{x}_{0:t})}{\int p(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}) p(\mathbf{x}_{0:t}) d\mathbf{x}_{0:t}} \tag{3.6}$$

A recursive formula for the joint distribution $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ is given by:

$$\begin{aligned} p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) &= p(\mathbf{x}_{0:t-1}, \mathbf{x}_t \mid \mathbf{y}_{1:t-1}, \mathbf{y}_t) \\ &= \frac{p(\mathbf{x}_{0:t-1}, \mathbf{x}_t, \mathbf{y}_{1:t-1}, \mathbf{y}_t)}{p(\mathbf{y}_{1:t-1}, \mathbf{y}_t)} \\ &= \frac{p(\mathbf{y}_t \mid \mathbf{x}_{0:t}, \mathbf{y}_{1:t-1}) p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t-1}) p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}) p(\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) p(\mathbf{y}_{1:t-1})} \\ &= p(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1}) \frac{p(\mathbf{y}_t \mid \mathbf{x}_{0:t}) p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1})}{p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})} \end{aligned} \tag{3.7}$$

In the last step in equation (3.7) one has $p(\mathbf{y}_t \mid \mathbf{x}_{0:t}, \mathbf{y}_{1:t-1}) = p(\mathbf{y}_t \mid \mathbf{x}_{0:t})$ due to the conditional independence of the observations given the states and $p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t-1}) = p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1})$ because the signal is a Markov process. In a Bayesian setting, the marginal distribution $p(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})$, can be computed recursively in two steps.

Prediction step: computed from the filtering distribution in using the law of total probability applied for conditional probability and the markov property of the process:

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1} \mathbf{y}_{1:t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}$$

In this step, due to the recursive equation (3.7), the filtering distribution $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1})$ is assumed to be known.

Update step: the prior is updated with the new measurement.

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t \mid \mathbf{x}_t) p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_t}$$

Despite the simplicity of those expressions, it turns out that to be computed they require the evaluation of high-dimensional integrals. A possible solution are Monte Carlo (MC) integrations methods, which are very powerful for evaluating high-dimensional integrals have the advantage to not impose any linearity or Gaussianity constraints to the model. Those methods have also very suitable general convergence properties.

3.4.2 The Monte Carlo approximation

Let us introduce the notion of particles for Monte Carlo computations, it is an independent and identically distributed (i.i.d) random sample. Let us assume that we can simulate N particles $\{\mathbf{x}_{0:t}^{(i)}, i = 1, \cdots, N\}$ according to the distribution $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$. An empirical estimate of this distribution denoted $\hat{p}_N(\cdot)$ is given by:

$$\hat{p}_N(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t}) \tag{3.8}$$

Where $\delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ denotes the delta-Dirac mass function located in $\mathbf{x}_{0:t}^{(i)}$. Here, $d\mathbf{x}_{0:t}$ can be viewed as an open set of $\chi^{(t+1)}$. Equation 3.8 defines an estimate of the distribution since it gives the value of the distribution for all open sets of $\chi^{(t+1)}$. The indice N is there to stress that the quality of the estimate depend on the number of simulated particles. Then, an estimate of $\mathbf{I}(f_t)$ is given by:

$$\hat{\mathbf{I}}_N(f_t) = \int f_t(\mathbf{x}_{0:t}) \hat{p}_N(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) \tag{3.9}$$

To obtain equation 3.9, one has used the fact that the integral and the sum can be exchanged as the sum is finite. Again the quality of this estimate depend on the number of simulated particles.

This estimate is unbiased and if the variance of f_t satisfies $\sigma_{f_t}^2 = \mathbf{E}_{p(\mathbf{x}_t|\mathbf{y}_{1:t})}[f_t^2] - I^2(f_t) < \infty$, then the variance of $\hat{\mathbf{I}}_N(f_t)$ is equal to $\mathrm{Var}(\hat{\mathbf{I}}_N(f_t)) = \frac{1}{N}\sigma_{f_t}^2$. From the strong law of large numbers :

$$\hat{\mathbf{I}}_N(f_t) \xrightarrow{a.s} \mathbf{I}(f_t), \text{for } N \to \infty$$

Moreover, if $\sigma_{f_t}^2 < +\infty$ then the Central Limit Theorem (CLT) can be applied :

$$\sqrt{N}[\hat{\mathbf{I}}_N(f_t) - \mathbf{I}(f_t)] \xrightarrow{dist} \mathcal{N}(0, \sigma_{f_t}^2), \text{for } N \to \infty$$

Thus from a set of particles $\{\mathbf{x}_{0:t}^{(i)} = 1, \cdots, N\}$ one can estimates any expectation $\mathbf{I}(f_t)$. The speed of convergence of this estimate is independent of the dimension of the integrand Newman, Barkema, and Newman [1999]. This is the main motivation for using Monte Carlo integration technique, since for any deterministic integration method the speed of convergence decreases as the dimensionality of the problem increase.

Unfortunately, it is usually hard to sample efficiently from $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ at any time t, as $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ is multivariate, non-Gausian, high-dimension and known up to a proportionality factor only. Monte Carlo Markov Chain (MCMC) is a popular approach to sample from a complex distribution Robert and Casella [2013]. Unfortunately, MCMC is unsuited for recursive estimation processes, because for every new observation a global estimation procedure have to be done. The modified importance sampling setting is well suited for a recursive procedure to make inference for HMM. But first let us introduce the *Importance sampling* method as an

intermediate step in the problem formulation.

3.4.3 Importance Sampling

The *importance sampling* method is an approximation used to estimate a particular distribution $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$, while only having samples generated from a different but related distribution, the so called *proposal distribution* $\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$. To be suited $\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ is required to have the same support, thus $\pi(\cdot) = 0$ implies $p(\cdot) = 0$. The proposal distribution may or may not depend on $\mathbf{y}_{1:t}$, in this last case it reduces to $\pi(\mathbf{x}_{0:t})$. Then an approximation of the expectation of f_t is given by:

$$\begin{aligned}\hat{\mathbf{I}}(f_t) &= \frac{\int f_t(\mathbf{x}_{0:t}) p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}} \\ &= \frac{\int f_t(\mathbf{x}_{0:t}) \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})} \pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}} \\ &= \frac{\int f_t(\mathbf{x}_{0:t}) w(\mathbf{x}_{0:t}) \pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}{\int w(\mathbf{x}_{0:t}) \pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) d\mathbf{x}_{0:t}}\end{aligned} \tag{3.10}$$

In equation (3.10) $w(\mathbf{x}_{0:t}) = \frac{p(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}|\mathbf{y}_{1:t})}$ is called the *importance weight*. The *normalized importance weight* $\tilde{w}_t^{(i)}$ are given by :

$$\tilde{w}_t^{(i)} = \frac{w(\mathbf{x}_{0:t}^{(i)})}{\sum_{i=1}^{N} w(\mathbf{x}_{0:t}^{(i)})}$$

Thus from a set of N particles $\{\mathbf{x}_{0:t}^{(i)} = 1, \cdots, N\}$ from $\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ a Monte Carlo estimate of $\mathbf{I}(f_t)$ is given by :

$$\begin{aligned}\hat{\mathbf{I}}(f_t) &= \frac{\frac{1}{N}\sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) w(\mathbf{x}_{0:t}^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w(\mathbf{x}_{0:t}^{(i)})} \\ &= \sum_{i=1}^{N} f_t(\mathbf{x}_{0:t}^{(i)}) \tilde{w}(\mathbf{x}_{0:t}^{(i)})\end{aligned} \tag{3.11}$$

It is interesting to note that, in equation (3.9) $\mathbf{x}_{0:t}$ are distributed according to $p(\cdot)$ and in equation (3.11) $\mathbf{x}_{0:t}$ are distributed according to $\pi(\cdot)$. Contrary to Monte Carlo approximation where it is required to know how to sample and evaluate the density of interest, in the importance sampling algorithm it is sufficient to evaluate the density of interest. Moreover, one usually know the distribution of interest only up to a constant. Indeed in equation 3.6 the integral in the denominator is hard to compute. It is only required to know how to evaluate a function proportional to $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) \propto p(\mathbf{y}_{1:t} \mid \mathbf{x}_{0:t}) p(\mathbf{x}_{0:t})$. Explicitly, $p(\cdot) = c \cdot \tilde{p}(\cdot)$:

$$\tilde{w}_t^{(i)} = \frac{\frac{c\cdot\tilde{p}(\cdot)}{\pi(\cdot)}}{\frac{\sum c\cdot\tilde{p}(\cdot)}{\pi(\cdot)}} = \frac{\frac{\tilde{p}(\cdot)}{\pi(\cdot)}}{\sum\frac{\tilde{p}(\cdot)}{\pi(\cdot)}}$$

Moreover the variance of the estimate can be lowered. Indeed in the Monte Carlo approximation the variance of the estimate is given by $\mathrm{Var}(\hat{\mathbf{I}}(f_t)) = \frac{1}{N}\sigma_{f_t}^2$ compared to the variance in the importance sampling framework $\mathrm{Var}(\hat{\mathbf{I}}(f_t)) = \frac{1}{N}\sigma_{f_t w}^2$. In using a smart choice of the proposal function then the weights will contribute to a reduction of this variance. In addition, it can be prove that there exist an optimal choice of the proposal distribution that optimize the mean square error [Sanz-González, Andina, and Seijas, 2002]. The Mean Square Error is given by MSE = bias^2 + var. It can be difficult to sample from it.

For N finite $\hat{\mathbf{I}}_N(f_t)$ is biased, as it is a ratio of estimates. However, under suitable assumptions, the strong law of large number applies, thus $\hat{\mathbf{I}}_N(f_t) \xrightarrow{a.s.} \mathbf{I}(f_t)$, for $N \to \infty$. Under additional suitable assumptions, due to CLT, the convergence rate is still independent of the dimension of the integrand Geweke [1989]. Importance sampling is a general Monte Carlo method. However, in this form, Importance sampling, does not allow for recursive estimation. One need to have all the data $\mathbf{y}_{1:t}$ available before estimate $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ or $\mathbf{I}(f_t)$. *Sequential Importance Sampling* is a strategy to overcome this problem.

3.4.4 Sequential Importance Sampling

The importance sampling method can be adapted in order to compute an estimate $\hat{p}_N(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ of the distribution without recomputing the trajectories of the particles $\{\mathbf{x}_{0:t}^{(i)}, i = 1, \cdots, N\}$ at each time a new observation is treated, but computing the estimate of the density based on previous estimate modified by the updated observation and signal. This algorithm is called *Sequential Importance Sampling* (SIS). This implies that the importance function $\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$ at time t admits as marginal distribution at time $t-1$ the importance function $\pi(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1})$:

$$\begin{aligned}
\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) &= \pi(\mathbf{x}_{0:t-1}, \mathbf{x}_t \mid \mathbf{y}_{1:t-1}, \mathbf{y}_t) \\
&= \frac{\pi(\mathbf{x}_{0:t-1}, \mathbf{x}_t, \mathbf{y}_{1:t-1}, \mathbf{y}_t)}{\pi(\mathbf{y}_{1:t-1}, \mathbf{y}_t)} \\
&= \frac{\pi(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})\pi(\mathbf{y}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t-1})\pi(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1})\pi(\mathbf{y}_{1:t-1})}{\pi(\mathbf{y}_{1:t-1}, \mathbf{y}_t)} \\
&= \pi(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1})\pi(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})\frac{\pi(\mathbf{y}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t-1})\pi(\mathbf{y}_{1:t-1})}{\pi(\mathbf{y}_t \mid \mathbf{y}_{1:t-1})\pi(\mathbf{y}_{1:t-1})} \\
&= \pi(\mathbf{x}_{0:t-1} \mid \mathbf{y}_{1:t-1})\pi(\mathbf{x}_t \mid \mathbf{x}_{0:t-1}, \mathbf{y}_{1:t})
\end{aligned}$$

Thus in iterating form:

$$\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \pi(\mathbf{x}_0)\prod_{k=1}^{t}\pi(\mathbf{x}_k \mid \mathbf{x}_{0:k-1}, \mathbf{y}_{1:k})$$

The importance weights can also be evaluated recursively:

$$
\begin{aligned}
\tilde{w}_t^{(i)} &= \frac{w(\mathbf{x}_{0:t}^{(i)})}{\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})} \\
&= \frac{1}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right]} \frac{p(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t}^{(i)} \mid \mathbf{y}_{1:t})} \\
&= \frac{1}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right]} \frac{p(\mathbf{x}_{0:t-1}^{(i)} \mid \mathbf{y}_{1:t-1}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})}{\pi(\mathbf{x}_{0:t-1}^{(i)} \mid \mathbf{y}_{1:t-1}) \pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \\
&= \frac{w(\mathbf{x}_{0:t-1}^{(i)})}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right]} \frac{p(\mathbf{x}_t^{(i)}, \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1}, \mathbf{y}_t)}{p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t}) \pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \\
&= \frac{w(\mathbf{x}_{0:t-1}^{(i)})}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right] p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1}) p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1})}{\pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \\
&= \frac{w(\mathbf{x}_{0:t-1}^{(i)}) p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1})}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right] p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})} \\
&= \frac{1}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right] p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})} w(\mathbf{x}_{0:t-1}^{(i)}) \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})} \\
&\propto \tilde{w}_{t-1}^{(i)} \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})}
\end{aligned}
\tag{3.12}
$$

In equation (3.12) $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}, \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1}) = p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ and $\frac{p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1})}{p(\mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t})} = \frac{1}{p(\mathbf{y}_t | \mathbf{x}_t^{(i)})}$ due to the conditional independence of the observations. $p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{0:t-1}^{(i)}, \mathbf{y}_{1:t-1}) = p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})$ because the signal is a Markov process independent of the observations. $\frac{1}{\left[\sum_{i=1}^N w(\mathbf{x}_{0:t}^{(i)})\right] p(\mathbf{y}_t | \mathbf{x}_t^{(i)})}$ is a constant. Finally, the SIS algorithm is, in general, a recursive application of the following update equations:

- $\mathbf{x}_t^{(i)} \sim \pi(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})$ (3.13)
- $\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)}) p(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)})}{\pi(\mathbf{x}_t^{(i)} \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{y}_{1:t})}$ (3.14)
- $\hat{p}_N(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^N \tilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ (3.15)

An important particular case is given when adopting the prior distribution as importance distribution:

$$
\pi(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = p(\mathbf{x}_{0:t}) = p(\mathbf{x}_0) \prod_{k=1}^t p(\mathbf{x}_k \mid \mathbf{x}_{k-1})
$$

In this case, the importance weights simplify to:

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$$

In the following, we will restrict the importance sampling distribution to the prior distribution. This choice is a very popular choice [Arulampalam et al., 2002]. Thus in this context, the SIS algorithm is given by the following distributions:

- $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$ (3.16)
- $\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ (3.17)
- $\hat{p}_N(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ (3.18)

The algorithm used in this thesis is described by the set of equations above. First one has to sample the prior distribution, then computing the update of the weights and finally estimate the distribution. The next two sections are devoted to computational complications.

3.4.5 Sequential Importance Resampling

In practice, iterating over equations (3.13) and (3.15) leads to the so-called *degeneracy problem*. As t increases, the distribution of the importance weight $\tilde{w}_t^{(i)}$ becomes skewed. In other words, only few particles will have a significant weight. Thus most of the computations to update particles trajectories will finally almost not contribute to the approximation of the density. This will make SIS fail to efficiently and effectively represents the distribution. The degeneracy problem is typically assessed by an estimate of the effective number of particles [Liu and Chen, 1998]:

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (\tilde{w}_t^{(i)})^2}$$

The key idea of the *Sequential Importance Resampling* (SIR) or *Bootstrap filter* algorithm is to introduce an additional selection step. In this chapter acronym SIR refer to Sequential Importance Resampling and not to the Susceptible-Infected-Recovered model. The purpose is to eliminate particles which have low weights and to multiply particles having high importance weight. Thus weighted empirical distribution $\hat{p}(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$ is replaced by the unweighted measure:

$$\hat{p}_N(d\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t}) = \frac{1}{N} \sum_{i=1}^{N} N_t^{(i)} \delta_{\mathbf{x}_{0:t}^{(i)}}(d\mathbf{x}_{0:t})$$

Where, $N_t^{(i)}$ is the number of offspring associated to particle $\mathbf{x}_{0:t}^{(i)}$ such that $\sum_{i=1}^{N} N_t^{(i)} = N$. Explicitly if $N_t^{(j)} = 0$ the particle j will die. $N_t^{(i)}$ have to be chosen such that the surviving

particles are approximately distributed according to $p(\mathbf{x}_{0:t} \mid \mathbf{y}_{1:t})$. There are many ways to select those particles. In practice this is done by N sampling steps with replacement over the particles [Gordon, Salmond, and Smith, 1993]. It is important to notice that after the resampling step, all the particles have the same weight. A schematic representation of the algorithm is presented in figure 3.4.

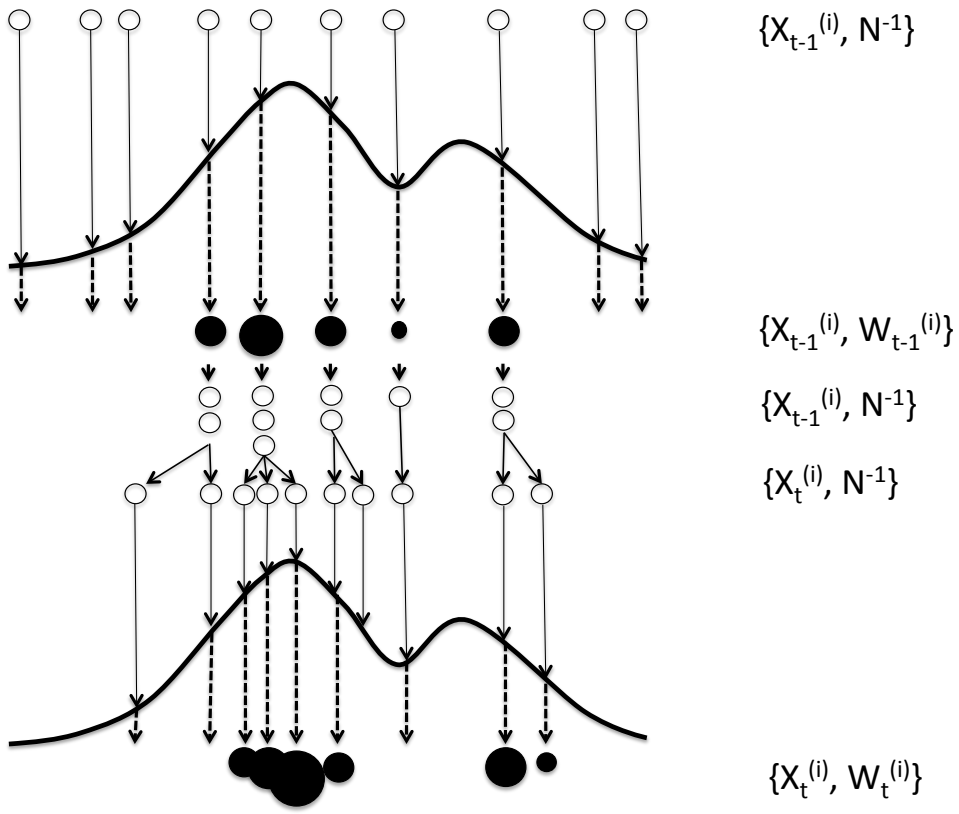

Figure 3.4: Schematic representation of the Sequential Importance Resampling (SIR) algorithm (adapted from [Doucet et al., 2001])

The SIR algorithm reduces to the same equations as the SIS and an additional resampling step. The main advantage of the SIR algorithm is that it only requires sampling from the distribution $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$, then evaluating $p(\mathbf{y}_t \mid \mathbf{x}_t^{(i)})$ and resampling. Due to the choice of the prior distribution, SIR leads to an importance distribution which is independent of the observation $\mathbf{y}_{0:t}$, and thus get rid of the information stored in the observations. In addition, addressing the degeneracy problem by removing the particles with small weights, resampling step introduces a new problem, called the *sample impoverishment problem*. As resampling the particle according to their weights, the diversity of the particles will tend to decrease. This problem is more severe for low noisy processes. In the extreme case, SIR can approximate the distribution with only one particle. In practice, in a SIR setting, to deal with *sample impoverishment problem* one can resample particles only if the estimated number of particles drop down to a certain cutoff [Arulampalam et al., 2002]. The problem stays as long as resampling step rely on a discrete distribution, the probability of choosing several time the same particle is not zero becoming

zero in the continuous case. Based on this ascertainment, *Regularized Particle Filter* Doucet et al. [2001] is a special SMC algorithm designed to address specifically this problem.

3.4.6 Regularized particle filter

As pointed out previously, the key idea of the Regularized Particle Filter (RPF) is to create a continuous distribution with the existing particles and to sample from this continuous distribution. The rest of the algorithm is the same as for SIR. In the RPF resampling step, the samples are drawn from:

$$\hat{p}_h(\mathbf{x}_t \mid \mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} \tilde{w}_t^{(i)} K_h(\mathbf{x}_t - \mathbf{x}_t^{(i)})$$

where, $K(\cdot)$ is the rescaled Kernel density. Here, $h > 0$ is a scalar parameter called the Kernel bandwidth and n_x is the dimension of the state vector $\mathbf{x}_t$. The rescaled Kernel density is defined as:

$$K_h(\mathbf{x}) = \frac{1}{h^{n_x}} K(\frac{\mathbf{x}}{h})$$

The most common optimality criterion used to select the bandwidth (h) is the Mean Integrated Squared Error (MISE). h is chosen to minimize the MISE between the true density and its estimate.

$$\text{MISE}(h) = \mathbf{E}\Big[\int [\hat{p}_h(\mathbf{x}_t \mid \mathbf{y}_{1:t}) - p(\mathbf{x}_t \mid \mathbf{y}_{1:t})]^2 d\mathbf{x}_t\Big]$$

In the particular case where all weights are equal, the optimal choice of the kernel density is given by the Epanechnikov kernel [Arulampalam et al., 2002]:

$$K_{Epa} = \frac{n_x + 2}{2c_{n_x}}(1 - \mathbf{x}^2)\mathbf{I}(|\mathbf{x}| \leq 1)$$

where c_{n_x} is the volume of the unit hypersphere in $\mathbb{R}^{n_x}$ and $\mathbf{I}(\cdot)$ is the indicator function.

Although the choice of the Epanechnikov kernel is optimal only in the case of equal weights, it can still be used in the general case to obtain a sub optimal approximation. The complexity of RPF is comparable to the complexity of the SIR. It only requires a finite sampling of a kernel density at each time step. However, RPF does not guarantee to asymptotically approximate the density. In practice, RPF performs better than SIR for low noise processes. Thus is described as the preferred choice SMC algorithm in [Arulampalam et al., 2002].

3.5 Parameter Inference for partially-observed nonlinear stochastic dynamical system

Parameter inference for partially-observed nonlinear stochastic dynamical system is known to be easier for time varying parameters [Ionides et al., 2006]. Indeed, due to the essence of the iterated filtering algorithm which applied sequentially, the maximization process arise at each step and not globally for the hidden states, parameter inference follows naturally the same scheme. In the Susceptible-Infected-Recovered context, the parameter of the model, namely the force of infection, the mean recovery period, birth and death rate are defined as time independent. The algorithm has to be adapted. A likelihood-based algorithm adapted to time independent parameter is presented below. The idea of this frequentist approach, called Maximum likelihood estimation via Iterated Filtering (MIF) [Bretó et al., 2009], is to replace the parameter by a slowly time varying random walk that converge to the maximum likelihood parameter estimate.

The general idea of performing parameter inference for a POMP object is to use a simulation algorithm, which should have the plug-and-play property to infer SIR models. Afterwards, one should wrap the simulator in a first SMC loop that give as output the distribution of the hidden states (which is called the filtering distribution). Then a second loop use a parameter inference algorithm such as MIF to find the maximum likelihood estimates of the parameter. Figure 3.5 shows a schematic representation of the big picture. A more rigorous description of the MIF is presented in 2.

3.5.1 Maximum likelihood via Iterated Filtering

This presentation follows Ionides et al. [2006] except that the derivations are extended in order to make them more comprehensible for non-specialist. Furthermore, additional steps are added. Maximum likelihood estimation via Iterated Filtering, called MIF, is a method that enable to perform maximum likelihood inference for a partially-observed nonlinear stochastic dynamical systems (also known as a POMP object). The method is based on a sequence of filtering operations which will converge to the maximum likelihood parameter estimate under certain conditions. Algorithms exist which handle efficiently time varying parameter (due to the natural sequential construction of the data). As the purpose of this thesis is to perform SIR inference the parameter are assumed to be time constant. Thus MIF algorithm is very convenient in this setting. In addition, maximum likelihood inference has various advantage: this is an efficient process, standard errors are available via the Hessian matrix and it is possible to perform model selection via maximum likelihood model comparison.

As described in the previous section a POMP object is completely specified by the conditional transition densities. The basic idea of the MIF algorithm is to replace time constant parameter $\boldsymbol{\theta}$ (could be multi-dimensional) by smoothly time varying parameter $\boldsymbol{\theta_t}$. Thus the conditional densities can be rewritten as:

- $p(\mathbf{x}_0, \boldsymbol{\theta}_0)$

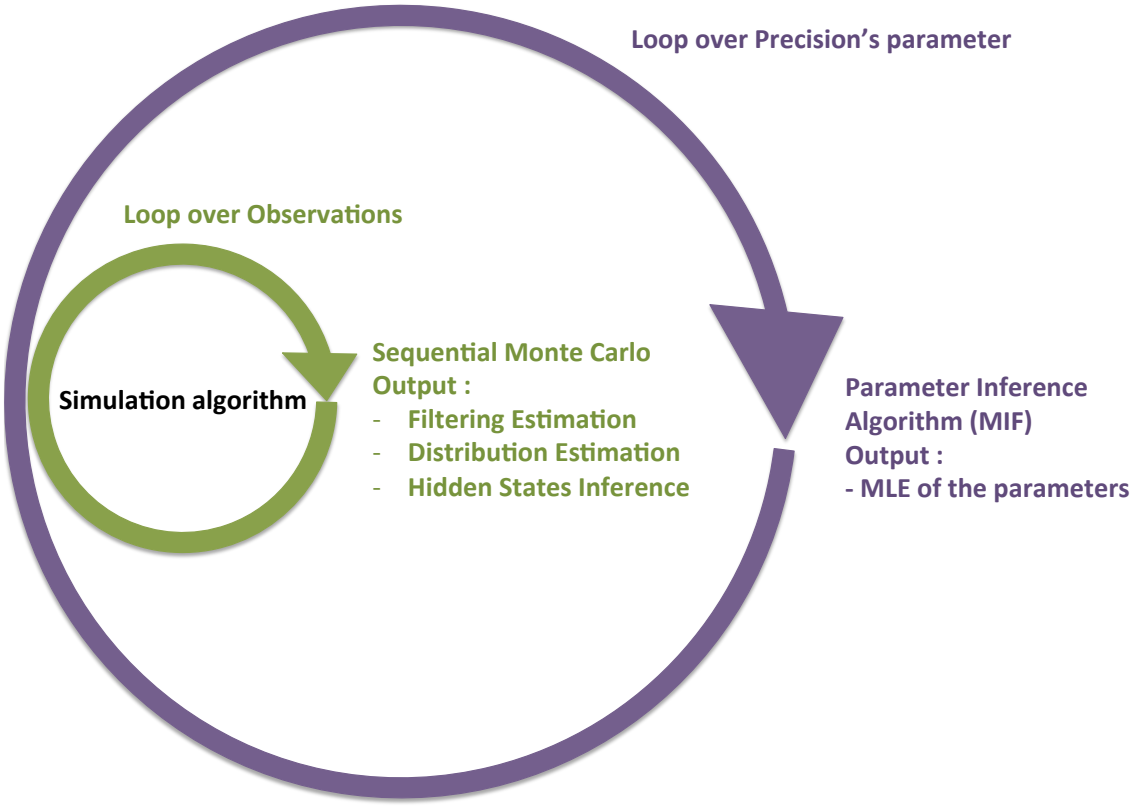

Figure 3.5: Schematic representation of an the global algorithm for POMP inference

- $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \boldsymbol{\theta}_{t-1}), \forall t \geq 1$
- $p(\mathbf{y}_t \mid \mathbf{x}_t, \boldsymbol{\theta}_t), \forall t \geq 1$

The time varying parameter $\boldsymbol{\theta_t}$ is a random walk:

- $\mathbf{E}[\theta_t \mid \theta_{t-1}] = \theta_{t-1}$
- $\mathbf{E}[\theta_0] = \theta$
- $\text{Var}(\theta_t \mid \theta_{t-1}) = \sigma^2 \Sigma$
- $\text{Var}(\theta_0) = \sigma^2 c^2 \Sigma$

where σ and c are scalar quantities.

3.5.2 Algorithm of the Maximum Likelihood Estimation via Iterated Filtering

The purpose is to get an estimate of the MLE for $\sigma \to 0$. The Maximum likelihood via Iterated Filtering algorithm is given below. It is an upper layer over the Sequential Monte Carlo (SMC)

estimation. Algorithm 2 presents the MIF procedure and the SMC procedure. Model inputs are defined by:

- $f(\cdot)$ is the density that generates the hidden states
- $g(\cdot)$ is the density that generates the observations
- $y_1, \cdots, y_N$ is the time series of the observations
- J is the number of particles of the SMC
- N is the fixed time lag
- M is the number of iterations of the MIF algorithm
- $0 < a < 1$ is called the cooling factor. Thus the variance reduction factor as the variance a^{m-1} for m$\rightarrow \infty$ is studied
- $b > 0$ is the scaling factor of the variance reduction factor
- $X_I^{(1)}$ is the initial state vector
- $\theta^{(0)}$ is the initial parameter vector
- Σ_I is the variance-covariance matrix of the initial state
- Σ_θ is the variance-covariance matrix of the parameter

Algorithm 2 Maximum likelihood Estimation via Iterated Filtering algorithm

for $m = 1$ to M **do**
  draw $X_I(t_0, j) \sim \mathcal{N}(X_I^{(m)}, a^{m-1}\Sigma_I), j = 1, \cdots, J$
  set $X_F(t_0, j) = X_I(t_0, j)$
  draw $\theta(t_0, j) \sim \mathcal{N}(\theta^{(m)}, ba^{m-1}\Sigma_\theta)$
  draw $\theta(t_0) = \theta^{(m)}$
  for $n = 1$ to N **do**
    set $X_P(t_n, j) = f(X_F(t_{n-1}, j), t_{n-1}, t_n, \theta(t_{n-1}, j), W)$
    set $w(n, j) = g(y_n \mid X_P(t_n, j), t_n, \theta(t_{n-1}, j))$
    draw $k_1, \cdots, k_J$ such that Prob$[k_j = i] = w(n, i)/\sum_l w(n, l)$
    set $X_F(t_n, j) = X_P(t_n, j)$
    set $X_I(t_n, j) = X_I(t_{n-1}, j)$
    draw $\theta(t_n, j) \sim \mathcal{N}(\theta(t_{n-1}, k_j), a^{m-1}(t_n - t_{n-1})\Sigma_\theta)$
    set $\theta_i(t_n)$ to be sample mean of $\{\theta_i(t_{n-1}, k_j), j = 1, \cdots, J\}$
    set $V_i(t_n)$ to be sample mean of $\{\theta_i(t_n, k_j), j = 1, \cdots, J\}$
  end for
  set $\hat{\theta}_i^{(m+1)} = \hat{\theta}_i^{(m)} + V_i(t_1)\sum_{n=1}^{N} V_i^{-1}(t_n)(\theta_i(t_n) - \theta_i(t_{n-1}))$
  set X_I^{m+1} to be the sample mean of $\{X_I(t_L, j), j = 1, \cdots, J\}$
end for

The output of the algorithm is:

- The maximum likelihood estimate for parameters $\hat{\theta} = \theta^{(M+1)}$
- The maximum likelihood estimate for the initial values $\hat{X}(t_0) = X_I^{(M+1)}$
- The maximum log likelihood estimate $\log \mathcal{L}(\hat{\theta}) = \sum_n \log(\sum_j w(n,j)/J)$

Here $\mathcal{N}(.,.)$ is the normal multivariate distribution. $X_I(t_n)$ takes values in $\mathbb{R}^{d_x}$, where d_x is the dimension of the hidden states. In the algorithm, F stands for filtering. The filtering distribution takes values in the same space. The observations y_i takes values in $\mathbb{R}^{d_y}$. The parameter θ takes values in $\mathbb{R}^{d_\theta}$ and has components $\{\theta_i; i = 1, \cdots, d_\theta\}$. Moreover $\theta_i(t_n)$ can be considered as local estimate of θ_i because they depend heavily on the observation around time t_n. As the number of iteration increase this dependence decrease. One can see that the steps 6-15 define an iterated filtering algorithm (or Sequential Monte Carlo algorithm). The updated estimate is a weighted average of the previous estimates. Theorem 7.3.1 in annex 7.3.1 show why using such a weighted average is a reasonable choice that approximate the maximum likelihood with respect to the parameters.

4 The pomp package model implementation

An R Core Team [2015] implementation of the Particle Filter and Maximum likelihood estimation via Iterated Filtering called pomp exists [King, Nguyen, and Ionides, 2014]. This section presents the pomp package [King, Ionides, Bretó, Ellner, Ferrari, Kendall, Lavine, Nguyen, Reuman, Wearing, and Wood, 2015a] and [King, Nguyen, and Ionides, 2015b]. This package provides tools to make inference for nonlinear partially-observed Markov processes. One can implement a model by specifying its hidden process and measurement components and then a broad range of method is implemented in order to make parameter inference.

In this thesis we use mainly two functions of this package: the particle filter algorithm (implemented as `pfilter()` function) and the maximum likelihood estimation via iterated filtering algorithm (implemented as `mif2()` function, which is an update of the algorithm presented in this thesis [Ionides, Bhadra, Atchadé, and King, 2011]). Several tutorials exist in order to get started with the pomp package (The web page (http://dept.stat.lsa.umich.edu/ ionides/tutorials/index.html) indexes all the tutorials about the pomp package).

The aim of this section is to clarify the link between the R implementation of the pomp package and the algorithmic methods presented in chapter 3. To do so, the code used to make a model implementation is presented and commented.

4.1 Model Implementation

In order to understand the implementation of the pomp package in the context described previously, a model implementation will be presented and commented. The core of the simulation algorithm is to define a POMP object. One has to use a so called POMP constructor, implemented as `pomp()`. This is an R function that encodes in a unique object: the conditional distributions that define the POMP, the time series of the data including the initial time, the names and initial values of the states and the parameters, the skeleton of the POMP (i.e set of differential equations that define the underlying SIR model), the function that generates the simulation, the possible covariates and the transformation of the parameter. It is often useful from a computational point of view, to transform restricted parameters to an unrestricted scale. Typically, the exponential-logarithmic function are used to extend real positive values and expit-

logit function to extend the $(0, 1)$ domain.

The advantage of the POMP constructor function is that it allows to specify the listed features using C code. Then, one has to use the `Csnippet()` function that allows to incorporate C code in R function. From a computational point of view this is more efficient.
The distribution of the hidden states is given by:

$$x_{i+1} \sim f(x_i, \boldsymbol{\theta})$$

The distribution of the observations is given by:

$$y_i \sim g(x_i, \boldsymbol{\theta})$$

The distribution $f(\cdot)$ contains state-update for the Markovian process and the distribution $g(\cdot)$ define for the emission process. The hidden likelihood of the system is given by:

$$\mathcal{L} = f(x_{0:T}, y_{1:T}, \boldsymbol{\theta}) = f(x_0, \boldsymbol{\theta}) \prod_{i=1}^{T} f(x_i \mid x_{i-1}, \boldsymbol{\theta}) g(y_i \mid x_i, \boldsymbol{\theta}) \tag{4.1}$$

where $\boldsymbol{\theta}$ is the parameter vector, the index i indicates the position in the time series and T is the total number of observations (i.e the length of the time series). To decompose the likelihood, we use the factorization described by the graphical model, see figure 3.1.

The hidden likelihood is quite complex. This justifies why an Expectation-Maximization (EM or Baum-Welch algorithm) approach for parameter inference would be untractable from a computational point of view. Indeed, the maximization step would require to maximize the hidden likelihood, thus computing all the possible hidden states $S(t), I(t)$ and $R(t)$ in the vector x_i and evaluating the likelihood. A priori, the hidden states are unbounded, therefore the hidden space is huge. This is why such approach is inefficient from a computational point of view. This is one of the rational to used Maximum Likelihood Estimation via iterated filtering.
In R, one constructs a POMP object, by specifying the following components:

- `rprocess()` : a simulator of the process model i.e a function to draw values from $f(x_{t_i} \mid x_{i-1}, \boldsymbol{\theta})$
- `dprocess()` : an evaluator of the process model, i.e a function yielding values of the probability density function $f(x_i \mid x_{i-1}, \boldsymbol{\theta})$
- `rmeasure()` : a simulator of the measurement model, i.e a function to draw values from $f(y_i \mid x_i, \boldsymbol{\theta})$
- `dmeasure()` : an evaluator of the measurement model, i.e a function yielding values of the probability density function $f(y_i \mid x_i, \boldsymbol{\theta})$

It is often not necessary nor possible to define for a particular model all those components. As a side remark, this explains why this thesis is called *Simulation based inference in epidemic models*. Indeed, oftentimes simulating a random processes is easier than evaluating their transition probabilities. From an algorithmic point of view it means that `rprocess()` is often easier to specify as `dprocess()`. Such approach is called a simulation based method, or alternatively plug-and-play, likelihood-free and equation-free method.

4.2 Model Definition for a single age strata SIR model

Figure 4.1 shows the states and transitions of a single age strata SIR model with demography. This model is specified by 4 compartments, 3 transition values, 2 additional parameters for the observation model and one parameter for the stochasticity. In the model μ is the birth and death rate of the population (no disease related), β is the force of infection, γ is the recovery rate. The emission process is defined by two parameters the size θ and the probability ρ. This model is an Euler multinomial. In this model definition H is the number of newly reported cases deduce from the total number of infected I. S is the number of susceptible and R is the number of recovered. Due to the model definition, the population stay constant even if individuals are only imported in compartments S. The specific death rate of the disease is neglected, which is a fair assumption for rotavirus, which has almost no extra mortality associated with it in industrialized countries.

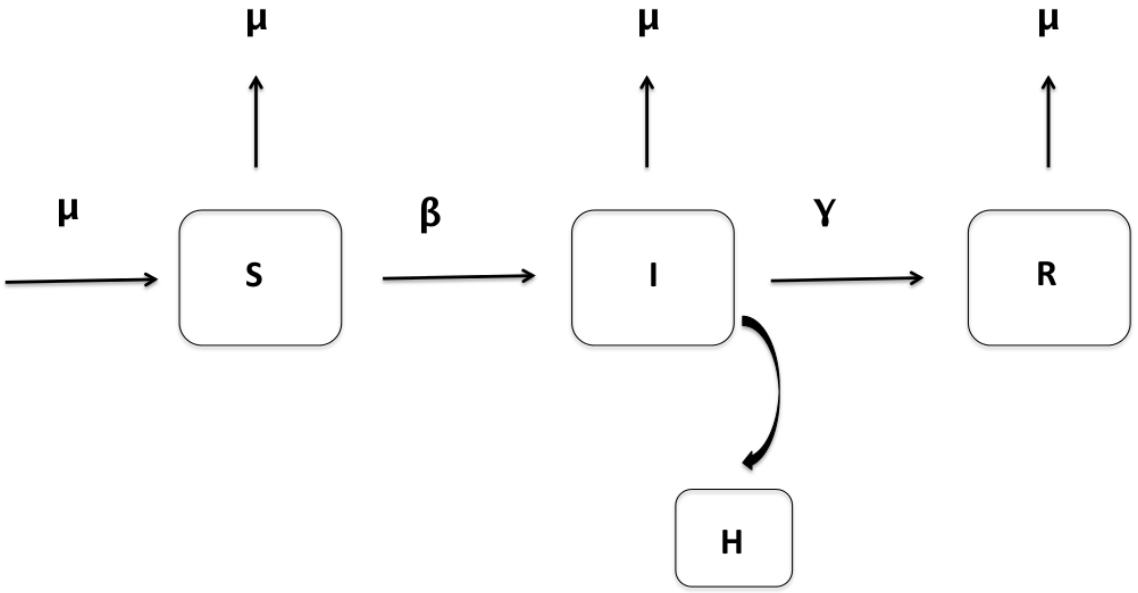

Figure 4.1: Schematic representation of an SIR model

The skeleton of the SIR model is defined by the following set of ODEs:

- $P(t) = S(t) + I(t) + R(t)$ (4.2)

- $\frac{dS(t)}{dt} = -\beta \frac{I(t)}{P(t)} S(t) - \mu S(t) + \mu P(t)$ (4.3)
- $\frac{dI(t)}{dt} = \beta \frac{I(t)}{P(t)} S(t) - \gamma I(t) - \mu I(t)$ (4.4)
- $\frac{dR(t)}{dt} = \gamma I(t) - \mu R(t)$ (4.5)

And the following emission process in discrete time:

- $H(t) \sim \text{NB}(\text{mean} = \theta\, i(t), \text{prob} = \rho)$ for $t = t_i$ (4.6)

The newly reported number of infected is a point wise (thus discrete time process) negative binomial distributed trial of the number of newly infected i.e : $i(t) = \beta S(t) I(t)/P(t)$. The negative binomial distribution is parametrize by two parameters: θ is called shape and ρ is called the probability. The stochastic representation of the model is then given by the following set of events and their respective rates:

Event	Rate
$(S(t), I(t)) \to (S(t) - 1, I(t) + 1)$	$\beta S(t) \frac{I(t)}{P(t)}$
$(S(t), I(t)) \to (S(t), I(t) - 1)$	$\gamma I(t)$
$(S(t), I(t)) \to (S(t) + 1, I(t) - 1)$	μ
$(S(t), I(t)) \to (S(t) + 1, I(t))$	μ

The emission process is given by the same discrete time process as in the deterministic setting.

Below, the R code used to generate this model is presented (the entire code is presented in 7.4). Here, the functions `rprocess()` and `dmeasure()` are defined in R in using a pomp constructor as:

```
rmeas <- "
cases = rnbinom_mu(theta, rho * i);
"
dmeas <- "
lik = dnbinom_mu(cases, theta, rho * i,give_log);
"
```

As one can see, the measurement process is negative binomial distributed with parameters θ for the size and $\rho \cdot i$ for the probability, where i is the number of newly infected. The argument *give_log* return the logarithm of the value of the function.

The SIR dynamic, defined by a multinomial model, is given by:

```
sir.step <- "
double rate[6];
double dN[6];
double dW; // white noise increment
double P;

P=S+I+R;

// compute the stochasticity
dW = rgammawn(sigma,dt);

rate[0] = mu * P;
rate[1] = (beta*I)/P*(dW / dt);
rate[2] = mu;
rate[3] = (gamma*dW)/dt;
rate[4] = mu;
rate[5] = mu;
dN[0] = rpois(rate[0] * dt);

reulermultinom(2, S, &rate[1], dt, &dN[1]);
reulermultinom(2, I, &rate[3], dt, &dN[3]);
reulermultinom(1, R, &rate[5], dt, &dN[5]);

S += dN[0] - dN[1] - dN[2];
I += dN[1] - dN[3] - dN[4];
R += dN[3] - dN[5];
i += dN[1];
"
```

As one can see the dynamics follow the equations given by table 4.2. The stochasticity is introduced in adding gamma noise to γ rate.

The initializer function is defined by following code. It can be seen that the relative initial distribution is specified by only one parameter and the initial number of infected $S(0)$ is set to a 1000 to reduce number of free parameters:

```
init <- "
S = popsize-1000;
I = 1000;
R = 0;
i = 0;
"
```

The skeleton of the SIR model is specified by:

```
sir.skel <- '
// transition rates
double rate[6];
// population size
double P;

// compute the transition rates
P=S+I+R;
rate[0] = mu * P;
rate[1] = beta * I/P;
rate[2] = mu;
rate[3] = gamma;
rate[4] = mu;
rate[5] = mu;

// assemble the differential equations
DS = rate[0]-rate[1]*S-rate[2]*S;
DI = rate[1]*S-rate[3]*I-rate[4]*I;
DR = rate[3]*I-rate[5]*R;
'
```

Parameter transformation are specified by:

```
fromEstimationScale <- "
Tgamma = exp(gamma);
Ttheta = exp(theta);
Tbeta = exp(beta);
Tmu = exp(mu);
Trho = expit(rho);
"

toEstimationScale <- "
Tgamma = log(gamma);
Tbeta = log(beta);
Ttheta = log(theta);
Tmu = log(mu);
Trho = logit(rho);
"
```

Finally, the pomp constructor that sums up everything is given by:

```
sir<-pomp( data = data.frame(cases = data,
                             time = seq(0, 51, by=1)),
           times = "time", t0 = -1/52,
```

```
            dmeasure = Csnippet(dmeas),
            rmeasure = Csnippet(rmeas),
            rprocess = euler.sim(step.fun =
                         Csnippet(sir.step), delta.t = 1/52/20),
            skeleton=Csnippet(sir.skel),
            skeleton.type='vectorfield',
            statenames = c("S", "I", "R", "i"),
            paramnames = c("gamma", "mu", "theta", "beta",
                           "popsize","rho"),
            zeronames=c("i"),
            fromEstimationScale = Csnippet(fromEstimationScale),
            toEstimationScale = Csnippet(toEstimationScale),
            initializer = Csnippet(init),
            params = c(popsize = 1000000, beta = 25, gamma = 10,
                       mu = 0.01, rho = 0.15, theta = 1)
)
```

The time series of the data is provided but also the vector of time points as a `data.frame()`. Here, the length of the time series is 52 time points, thus is weekly reported data over a year. The start of the time series is set one time step before starting of the time series and given by `t0`. The `rprocess()` is generated in using a function, `Euler.sim()` which produce an Euler scheme. The simulation time step of the process is set to twenty times the frequency of the time series of the data. Based on those components, a pomp object is created and the features described in the next chapter could be applied on.

4.3 Functionality of the pomp package

The following features have been developed in the `pomp` package. Many other feature have also been developed but are not presented here as they are not used in this current work.

- Simulate a model using `simulate()`
- Integrate your model's deterministic skeleton using `trajectory()`, here skeleton mean the deterministic part of the model
- Estimate the likelihood of a model for a given set of parameters using Sequential Monte Carlo, implemented in `pfilter()`
- Find the Maximum Likelihood Estimates for parameters via iterated filtering, implemented in `mif2()`

4.4 Deterministic Model Implementation using `deSolve` and `pomp` R package comparison

The purpose of this section is to exemplify and asses the model implementation, presented in figure 4.1, in a deterministic setting. In figure 4.2, shows a model implementation and solutions using both `deSolve` and `pomp` R packages. As one can see there is a great agreement between those very different implementations. The model parameters used are:

- $\beta = 0.5$
- $\mu = 1e^{-4}$
- $\gamma = 0.005$

The set of initial conditions are given by:

- $P(0) : 1000000$
- $S(0) = 999999$
- $I(0) = 1$
- $R(0) = 0$

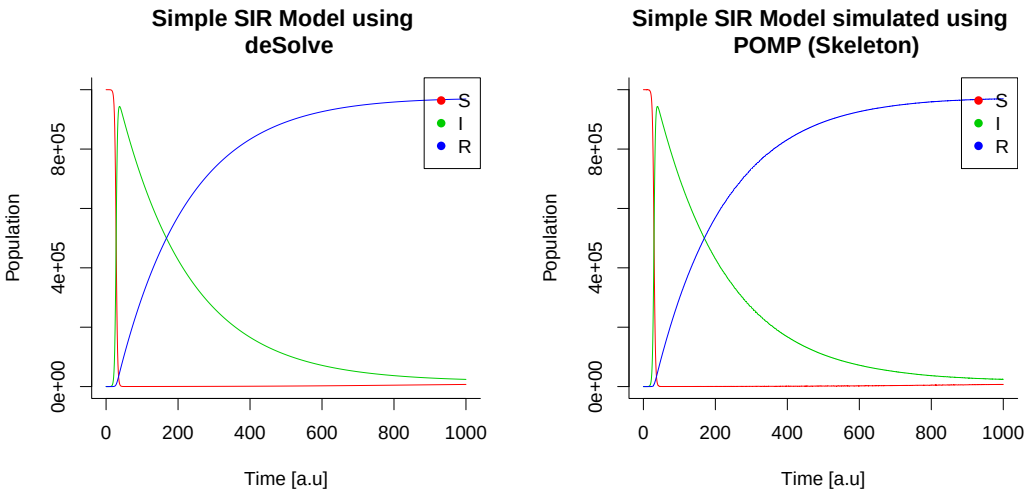

Figure 4.2: Solving dynamical SIR System using deterministic simulations

The set of initial values and parameters produce a single outbreak and the trajectory can be seen in figure 4.2. The implementation using deSolve is based on solving the ODE model using the `deSolve` package. The skeleton implementation is the deterministic implementation of the `pomp` package. Both graphs are very similar.

4.5 Stochastic Model Implementation using pomp R package

Using the Euler Multinomial with Gamma noise algorithm (see algorithm 1 on page 23) implemented in the pomp package framework gives figure 4.3. The intensity of the gamma noise has been chosen to be $\sigma = 8$.

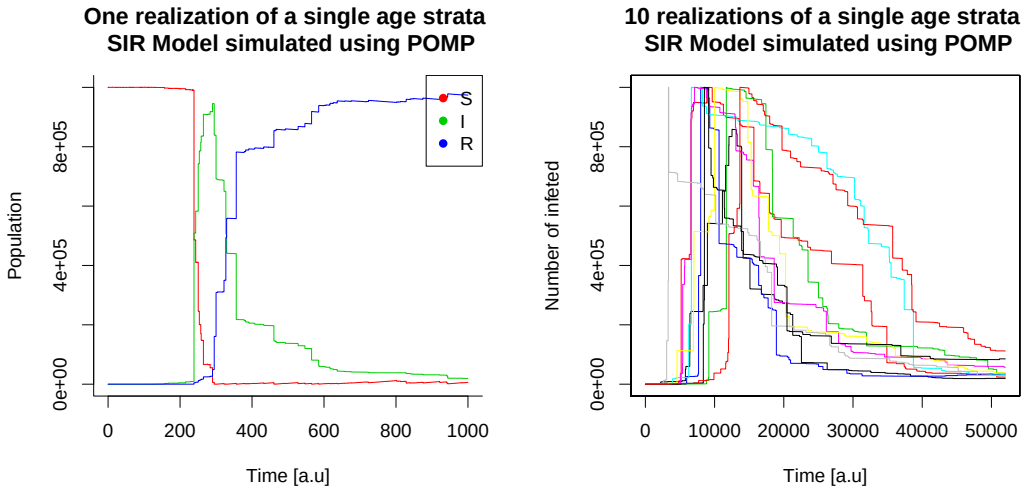

Figure 4.3: Euler Multinomial with Gamma noise applied to a single age strata with demography

The implementation of this model gives coherent results. Indeed the stochasticity seems very similar to figure 3.3.

4.6 Remarks about parameter inference using pomp package

All computation presented in this thesis have been done using R Core Team [2015]. Generally pomp computations are heavy. Even after using C code in the pomp constructor, the computations runs have to be written in parallel. To parallelize the code used in this thesis multiple **R** packages `foreach` [Analytics and Weston, 2014a], `doMC` [Analytics, 2014], `parallel` [R Core Team, 2015] and `doParallel`[Analytics and Weston, 2014b].
Moreover, heavy pomp computations are best performed in parallel on a cluster or multi-core machine. Thus all computations have been performed on a 8 cores cluster virtual machine from the Zurich University. Some challenges arise to ensure reproducibility and more important to avoid useless computations. To ensure reproducibility, seeding have been done using l'Ecuyer L'ecuyer, Simard, Chen, and Kelton [2002] technique. Avoiding repetition of expensive calculations have been done in using `stew()` function of the pomp package.

5 Results

This chapter described the results obtained in using the `pomp` package to perform simulation based inference applied to the rotavirus reported data. As a proof of concept, a simulation study is presented to exemplify the inference method used to find the maximum likelihood estimate (MLE) of an SIR model.

The overall purpose is to find the MLE for a single age strata model without seasonality with and without gamma noise. Then to compare the impact of the gamma noise on the model's predictions. It turns out that the gamma noise is a very sensitive parameter for inference procedure. Then as a first step, a model without gamma noise will be studied, then the MLE of this model will be used to infer the model that include extra noise. The data used are the data presented in chapter 2 for year 2002-2003 (see figure 2.3, red line).
A single age SIR model without seasonality with demography with gamma noise is defined by 7 model parameters (3 parameters for the model, one for the population, two for the emission process and one for the stochasticity). The gamma noise, also called extra noise, is determined by a unique parameter.

5.1 Simulation study

A simulation study have been performed in order to assess the validity and the efficiency of the algorithm using the model defined in previous chapter (see figure 4.1). The model used is a single age strata without seasonality without gamma noise (to reduce the computation workload). First the model is used to simulate some data, then those data are used to infer the model's parameters.

In the simulation study, the parameter popsize is set to 80,000,000 to be in line with the German population. The parameter μ is set to 0.00015 as an estimate of the mean birth rate in Germany in 2002 [of Statistics, 2013]. The model's parameters are set to $\rho = 0.5$, $\beta = 5$, $\gamma = 5$ and $\theta = 50$. Figure 5.1 shows the data simulated used to perform this simulation study.

In order to find the MLE of the model's parameters, the following procedure will be applied. As a first step a so called local search is performed, therefore the prior value of the MIF

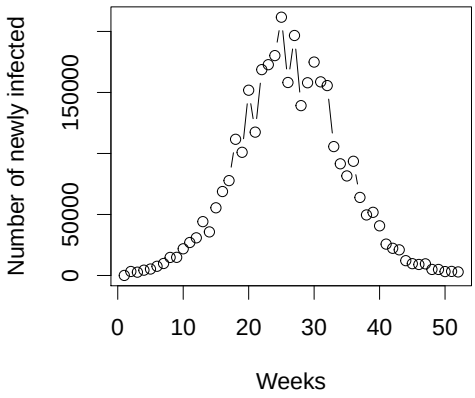

Figure 5.1: One realization of the number of newly infected for (beta=5, gamma=5, theta=50, rho=0.5, popsize=80,000,000, mu=0.00015)

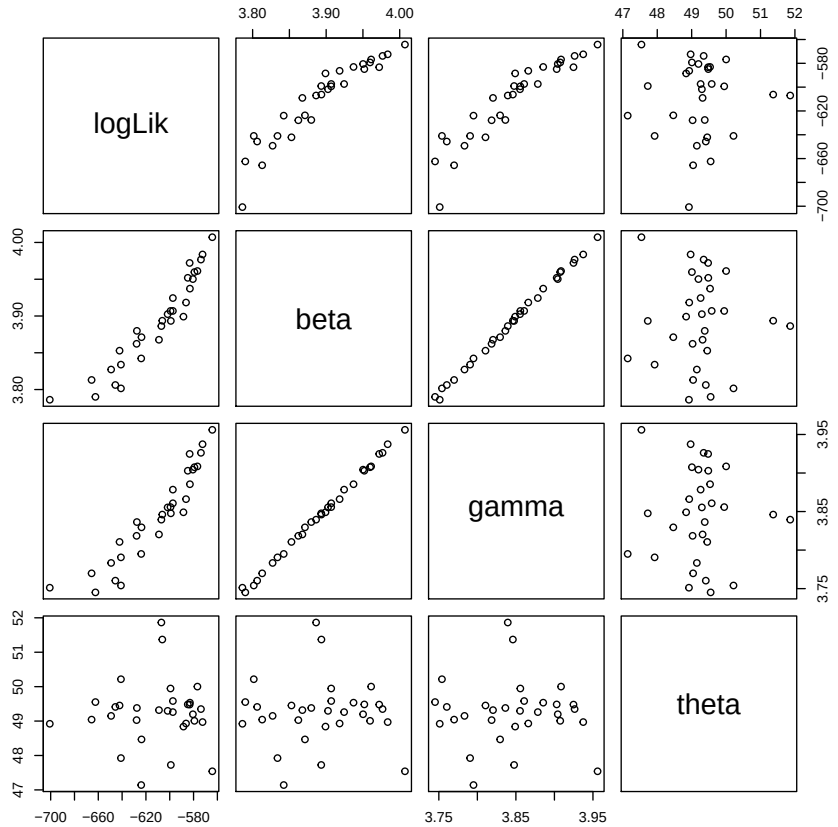

Figure 5.2: Pair plots of the log likelihood with the parameters for a simulation study Simulation parameters (Np=10000; Nmif=100; Nlocal=31/40)

algorithm is always the same for all simulations. In this simulation study the true parameter values will be taken as prior values. Then a global search is performed, i.e starting values of the MIF algorithm will have a distribution. This step is necessary as even theorem 7.3.1 assures the global convergence of the algorithm and then suggest a local behavior, is computationally valid only for an infinite computation time [Liu, Chen, and Logvinenko, 2001] and [Spall, 2003]. Thenceforth several starting values have to be tested, this is called the global search. Usually this step requires a large computation time.

Figure 5.2 shows the pair plot graph of the model's parameter in function of the log likelihood of the local search. It can be seen on figure 5.2, that β and γ are highly correlated. Moreover, those parameters are also strongly correlated with the log likelihood. Figure 5.3 shows the trajectories of the model's parameters for the local search (same starting value) and global search (distribution of starting value). On can see that the MIF algorithm fails to find the MLE for β and γ.

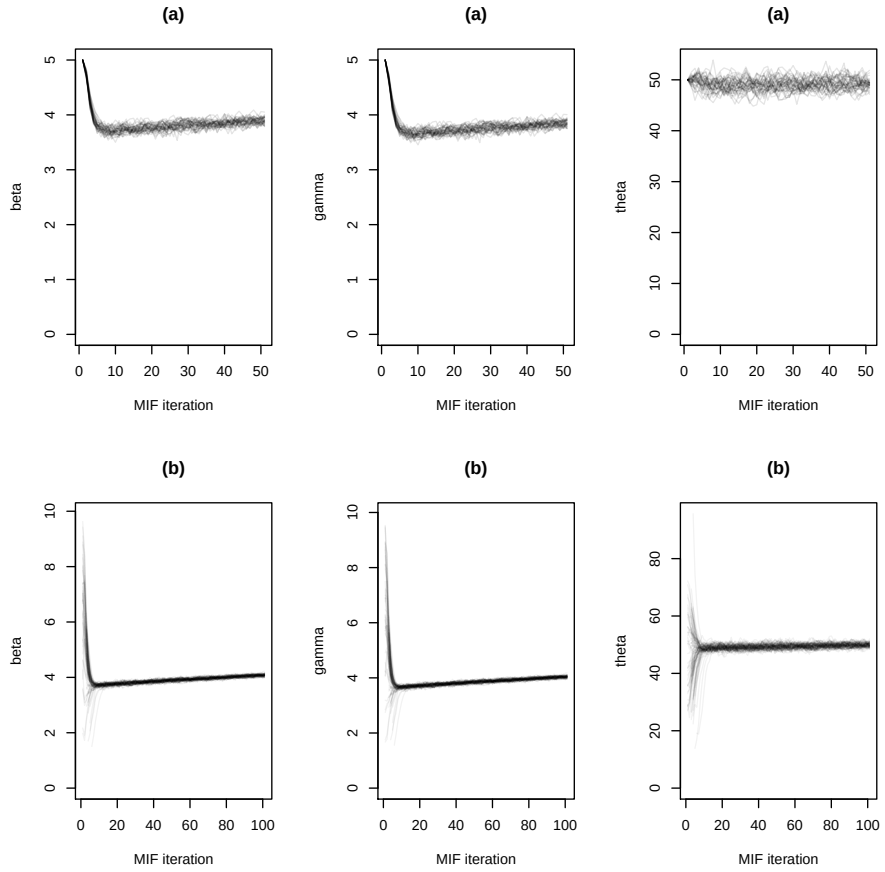

Figure 5.3: Trajectories of the model parameters (beta,gamma,theta) for a simulation study for local (a) and global (b) searching procedure

The fact that β and γ are strongly correlated can be observed on both local and global search. This is due to the fact that this simulation study infers model's parameters based on the newly reported number of cases. In the model used, the number of newly cases at each time is

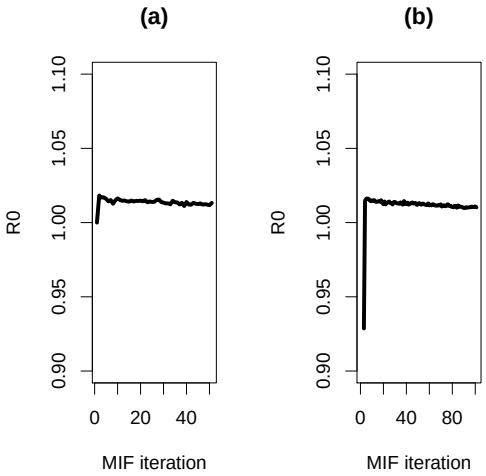

Figure 5.4: R0 estimation for local (a) and global (b) search

mainly driven by the relative value of β and γ. In other words, there is a deal of information between $R_0 = \beta/\gamma$, which gives the final size of the epidemic, and the individual values of β and γ. From a geometrical point of view, there is a ridge in the parameter space with a sharp curvature in the R_0 direction and a smooth curvature in complementary direction. This explain why it is relatively easy to find an estimate of R_0 and computationally hard to find a good estimate of β and γ individually. Figure 5.4 shows the plot of R_0 in function of the MIF iteration for the local and global search. As one can see, both values are very close to the true value ($R_0 = \beta/\gamma = 5/5 = 1$).

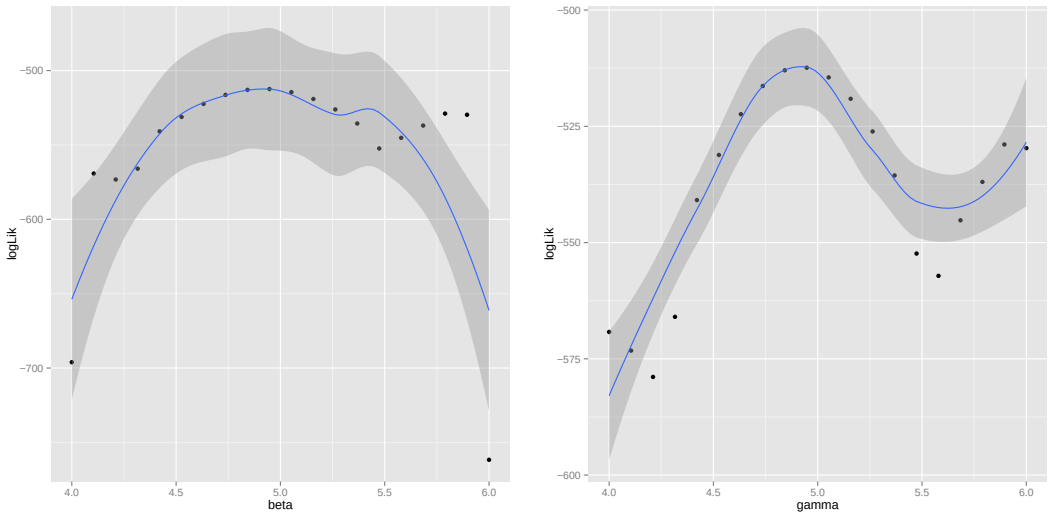

Figure 5.5: Likelihood profiles for gamma and beta

To overcome this issue, we will use the likelihood profiles. Figure 5.5 shows an estimate of the likelihood profiles for β and γ optimized together. As one can see on this graph, the maximum likelihood is estimated for $\beta = 4.9$ and $\gamma = 4.9$ which is very close to the true value

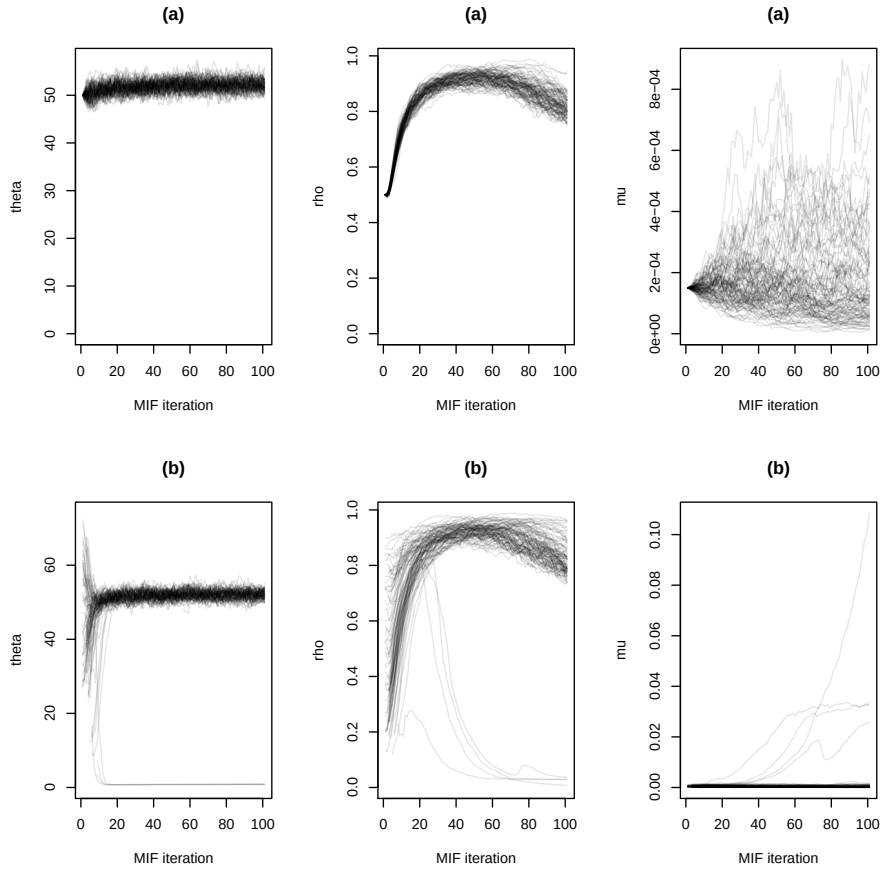

Figure 5.6: Trajectories of the model parameters (theta,rho,mu) for a simulation study for local (a) and global (b) searching procedure

of those parameters. To construct the likelihood profiles, the true values have been used for other model's parameters and a grid of sampled value of β and γ where the likelihood have to be evaluated. To get a reliable estimate each points is computed 15 times, then only maximal values are kept and plotted. On figure 5.5 blue line is a loess estimate of the profiles likelihood and grey surface is an estimate of the computational variance of the profile likelihood. Even in this highly favorable situation (known true value), the computational efforts to get the estimate of the profile likelihood are heavy.

Figure 5.7 shows an other estimate of the profile likelihood of the model's parameters. As one can see, the profile likelihood for β, γ is very sharp. It can also be observed that ρ has a no symmetric profile likelihood. This is an indication why MIF algorithm overestimates it. Finally, μ seems to be not very well identifiable due to the large curvature of the profile likelihood.

An important remark is that figures 5.5 and 5.7 present two estimates of the profile likelihood but are very different in essence. Indeed, figure 5.7 can only be computed in knowing the MLE of all model's parameters. Indeed, each graph is obtained in using MLE value of all parameters except the one varying. Whereas 5.5 can be computed without previous knowledge of the MLE, and thus can be use in more general inference context, but is computationally more

demanding.

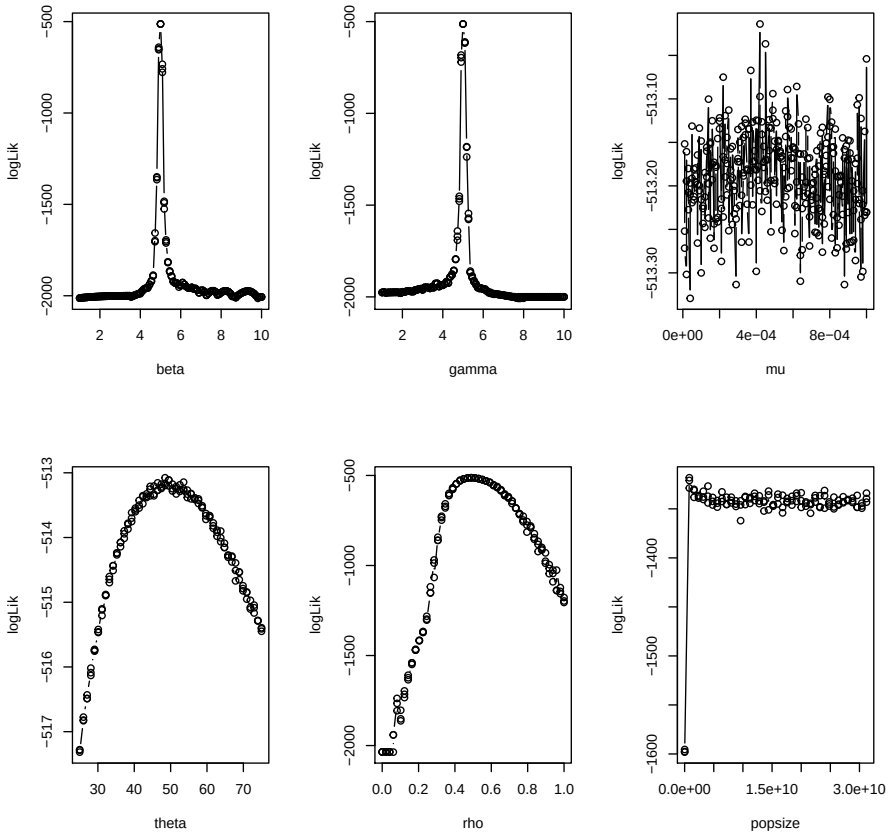

Figure 5.7: Likelihood Profiles

The MIF estimation of the other model's parameters, except $popsize$, are presented below. Indeed, figure 5.6 presents the the trajectories of the model's parameters for the local search (same starting value) and global search (distribution of starting value). Contrary to the previous trajectory plots, it can be observed that the trajectory plot of the global search contains some ouliers whereas no outliers are observed in the local search. This could be due to the fact that to obtain this graph all parameters are under optimization whereas in previous search only β, γ and θ were optimized.

Figure 5.8 shows the vioplot plots of the distributions of θ, ρ, μ. A vioplot is a convenient way to present a distribution. It is a boxplot with a kernel density estimate. The white dot is the median, the borders of the black box is the first and the third quartiles and the black line is 1.5 Inter Quartile Range (IQR)). The blue curve is a kernel density estimate. Figure 5.8 presents the distribution of the model's parameters at the last MIF step of the local and global search presented in figure 5.6. As one can see, on the vioplots the outliers are visible in the distribution of the parameters in the global search. An open question is which estimator has to be used to best estimate the MLE. Indeed, the mean, median and mode of the distribution are quite different. Based on this estimation of the distributions of the model's parameters, table 5.1 shows the

mean, median and mode estimates and the relative errors to the true values. The relative error is defined as the difference between the true value and the estimate normalized by the true value. It can be observed, that the mode seems to best estimate the MLE for the model's parameters.

The fact that MIF procedure give us access to the distribution of the parameter is very interesting. Indeed, model's predictions can be computed in sampling the distribution of the parameters and in simulating realizations of the model. Using a large number of realizations and quantile regression it is possible to estimate a confidence region for the model's prediction. It is very interesting because, this estimation do not assume any gaussianity of the parameter's distribution. Assessment of uncertainty of epidemic model is a challenge because most of the time prediction are based on point estimation of the model's parameters and not the distribution.

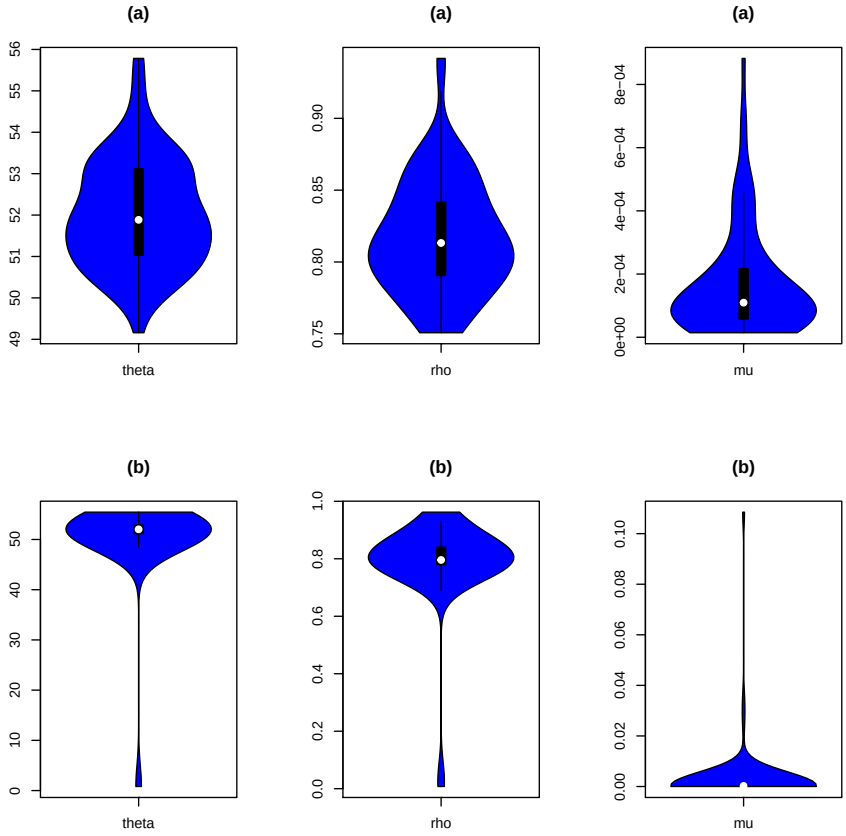

Figure 5.8: Vioplots of the model's parameters (mu,rho,theta) for a simulation study for local (a) and global (b) searching procedure

Table 5.1: True value and estimates of the model's parameters

Parameter	True Value	Mean	RR [%]	Median	RR [%]	Mode	RR [%]
Local search estimates							
μ	0.00015	1.66e-04	-10	4.48e-04	-198	7.44e-05	50
ρ	0.5	8.17e-01	-63	8.46e-01	-69	8.04e-01	-12
θ	50	5.20e+01	-04	5.24e+01	-05	5.16e+01	-03
Global search estimates							
μ	0.00015	2.31e-03	-1440	5.42e-02	-36033	2.79e-04	-86
ρ	0.5	7.90e-01	-58	4.84e-01	3	7.87e-01	-57
θ	50	5.02e+01	-0.4	2.81e+01	44	5.21e+01	-4

5.2 Simulation based inference investigations using pomp package with a single age strata SIR model without gamma noise applied to rotavirus data

In this section and the next one, all computations have been done using 5'000 particles. The simulated model used has a frequency 20 times higher than the frequency of the observed time series. The magnitude of the random walk of the MIF algorithm is set to 0.02. Final, a geometric cooling schedule, fixed at 0.8, has been used.

In order to find the MLE of a single age strata SIR model without gamma noise applied to rotavirus reported data a local search has been done. Figure 5.9 shows the pair plot of the model's parameters and the log likelihood for 300 MIF runs. The log likelihood distribution have a long tail for high values with respect to all parameters, this is an indication that convergence is not achieved. One can also see that γ and β are highly correlated which is a major issue for the optimization process. This problem has already been encountered.

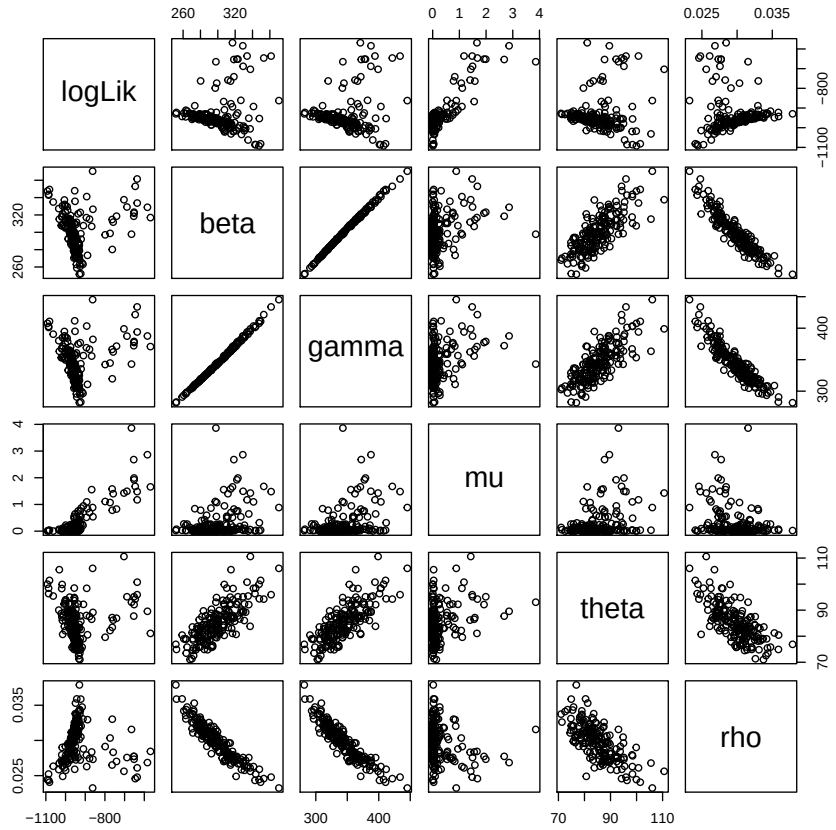

Figure 5.9: Pair plots of the log likelihood with the model's parameters

Figure 5.10 shows the model's parameters trajectories for 300 MIF iterations (μ is presented two times for rescaling purpose). As one can see the pattern of convergence depends heavily

on the parameter. For example, the parameters β, γ and ρ seem to be correlated. Thus convergence is not achieved. Whereas θ seems to converge. Here, μ has a very broad distribution. A possible explanation is that low values of μ gives a birth rate that only "shift" the bell curve of the number of newly infected.

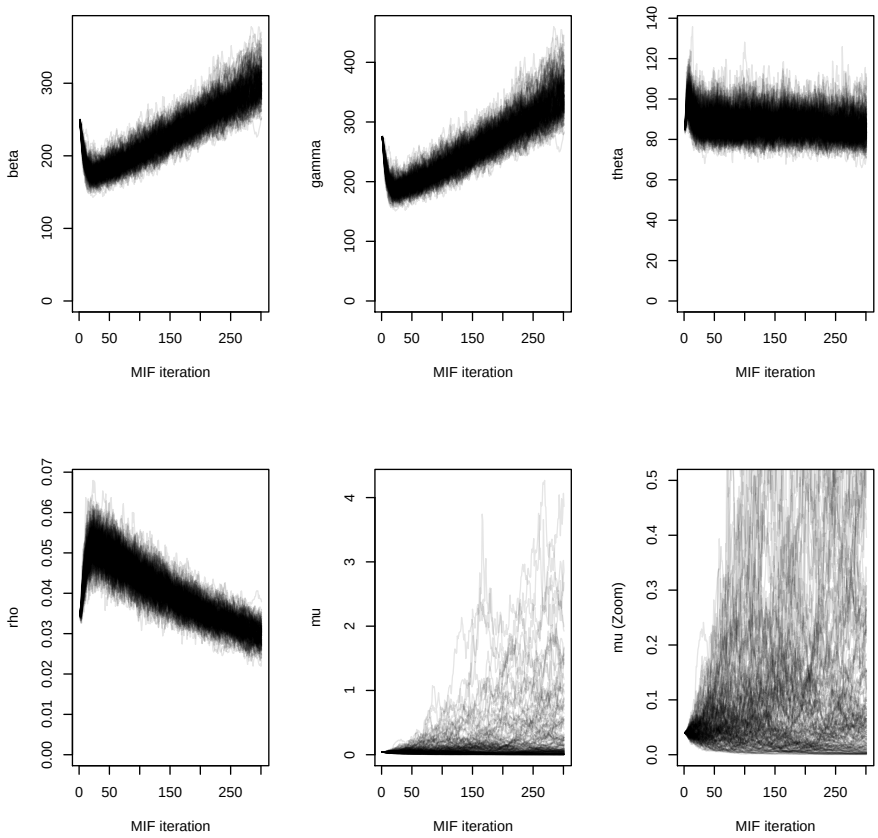

Figure 5.10: Trajectories of the model parameters

Based on those preliminary results and the simulation study, the idea is that convergence is not achieved because there is a deal of information between β and γ. In contrast to the previous section the MLE is unknown here. Then a blind search following the ridge between β and γ is computationally intractable. There is a need for a good proposal sets of values to be tested.

As previously pointed out, in a single outbreak context there tends to be a deal of information on $R_0 = \beta/\gamma$, which determines the final size of the epidemic and the invasion speed. Thus β and γ are hard to be estimated individually but R_0 seems to be well estimated. Figure 5.11 shows R_0 in function of the MIF iteration. As one can see the variations is pretty small, mean value is 0.91.

In order to try to infer β and γ individually we will again use the profiles likelihood. Another possible solution is to use the information of several outbreaks. But due to time constrain this solution is not possible. The idea is to fit the values of the mean of the parameter distri-

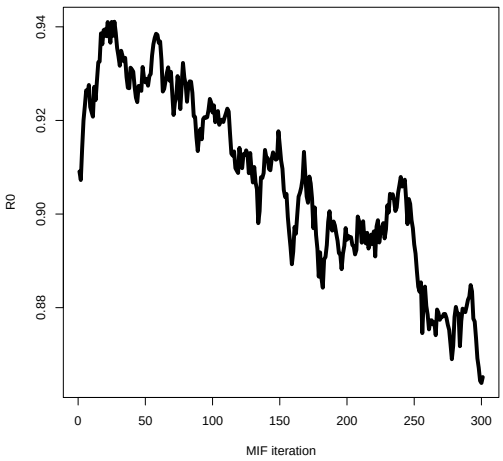

Figure 5.11: R0 estimation

Table 5.2: Model's parameters estimation in function of the number of MIF iteration

MIF iteration	β	γ	θ	ρ	μ
100	204.71	222.50	87.77	0.043	0.087
150	226.15	248.83	87.06	0.039	0.136
200	249.49	278.16	87.35	0.035	0.174
250	273.44	309.38	85.86	0.032	0.229
300	299.11	343.73	85.57	0.029	0.295

bution for a given number of MIF iteration in order to find their relationship. To compute the profile likelihood for sampled parameters values that fulfills the relationship and to keep the set of parameters that leads to the maximal value of the log likelihood. The estimation of θ, ρ and μ seem to be easier. The mean of the distribution of θ and ρ seem more or less independent of the number of MIF run.

Based on the parameters estimations during the 300 MIF runs, table 5.2 shows the mean of the model's parameters for a selected number of MIF iterations.

A linear relation between β and γ has been assumed and fitted (Intercept = 32.2, slope = 0.78). Again the purpose is not to find the MLE using this technique but only some proposal sets of model's parameters values to be tested in order to compute more efficiently the profile likelihood. Unfortunately, this method do not guarantee to find the global maximum of the log likelihood.

Figure 5.12 shows the profile likelihood in function of β. The largest log likelihood is achieved for the following set of parameters $popsize = 80000000, \beta = 401.6667, \gamma = 498.8889, \mu = 0.03, \rho = 0.0175, \theta = 75$.

Figure 5.13 presents the reported data used to fit the model with 1000 realizations of the model using the MLE and 97.5% and 2.5% confidence bands computed with a quantile regres-

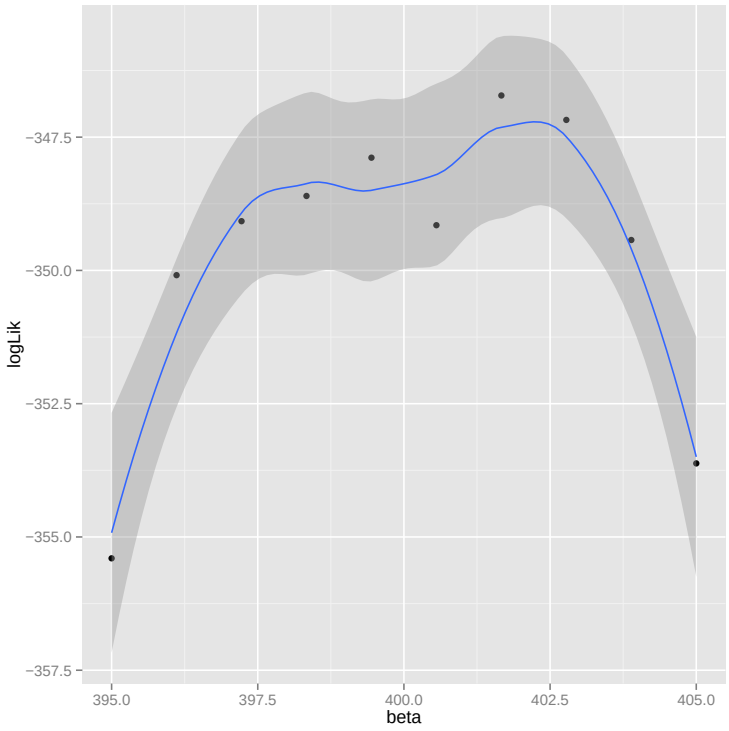

Figure 5.12: Profile likelihood of a single age strata without gamma noise

sion. As one can see, the model's predictions do not succeed to reproduce the stochasticity of the data. Thus additional stochasticity has to be added. Next section investigates the adding of gamma noise.

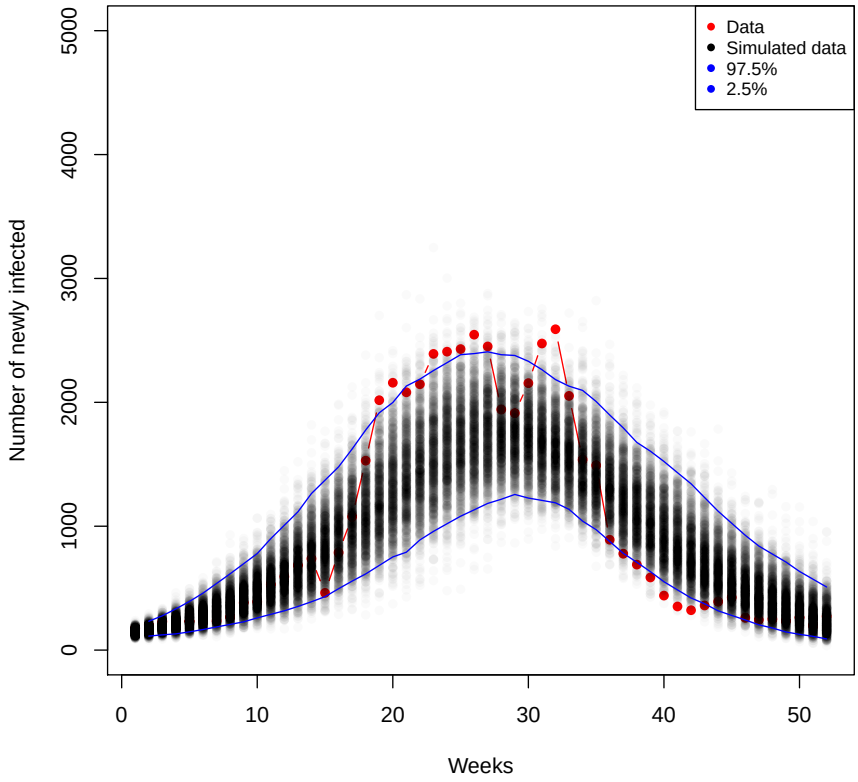

Figure 5.13: Simulations predictions using MLE

5.3 Simulation based inference investigations using pomp package with a single age strata SIR model with gamma noise applied to rotavirus reported data

This section investigates the adding of gamma noise to the model described in the previous section. The idea is to use the MLE found previously and using the MIF algorithm to find how much extra stochasticity is implied by the data. The gamma noise is a very sensible parameter for the optimization then it is treated separately.

Based on the MLE previously found, a local search of sigma has been done. Figure 5.14 presents both the trajectory plot and the vioplot of σ for a local search. As one can see the convergence seems to be achieved. It can be observed on the vioplot that the kernel density estimate of the mean (5.81e-03), the median (5.97e-03) and the mode (5.73e-03) are not very different.

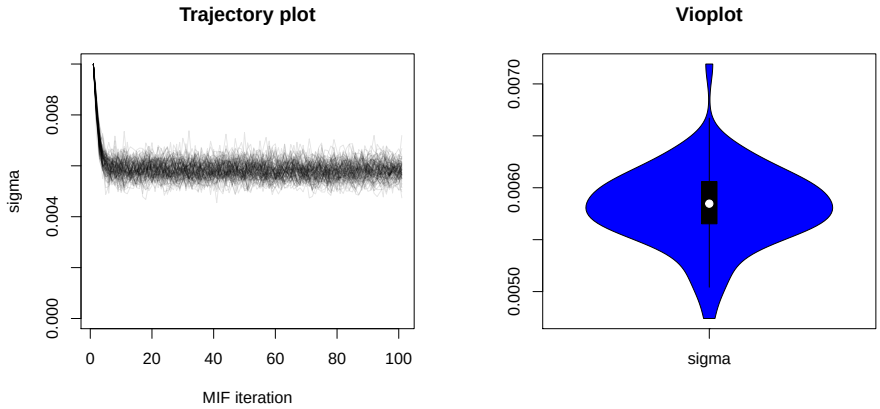

Figure 5.14: MIF iterations of gamma noise parameter

By way of example, a comparison between full model's prediction and the rotavirus data is presented. Using the mean estimate for σ and the MLE computed in the previous section, figure 5.15 the model's predictions and the weekly reported data from 2001-2009. The model's parameters have been inferred using 2002 data only. The confidence bands have been computed using a quantile regression (97.5% and 2.5 %) of 10'000 simulations of the full model. As one can see the coverage of the weekly reported data by the simulated data is better with gamma noise. However, the model's prediction seem to fail to accurately find the peak of the epidemic which is in March. The model's peak prediction is the 7th week of the year (February) for the upper confidence band and the 11th week (March) for the lower confidence band. Moreover, the model's prediction of the beginning of the epidemic is inaccurate, indeed confidence region starts to early compared to the data. The end of the epidemic seems contrariwise better fitted. The beginning and end of an epidemic can be defined as threshold number of case for example. Then further computations and investigations are needed to improve the model's predictions.

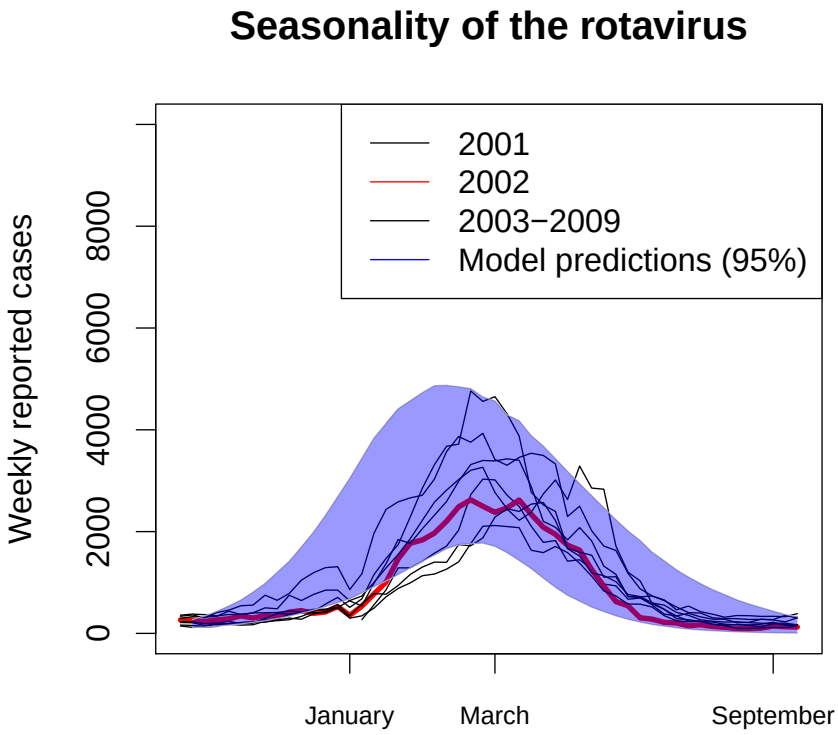

Figure 5.15: Model's simulations using MLE

6 Conclusion

The Particle filter and maximum likelihood via iterated filtering algorithms have been presented. Also applications of those algorithms to simulation based inference on SIR epidemic models. Some mathematical derivations have been extended to highlight the challenges to make parameter inference for complex problems. The algorithmic implementations of the Particle filter and Maximum Likelihood via iterated filtering have been described. Some of the computational issues have been shown and discussed. The `pomp` package have been presented. The main features of the package have been described.

As an example, a single age strata without seasonality without gamma noise have been implemented and studied in a simulation study. It has been possible to accurately infer the MLE of the model used. Nonetheless some issues have been reported and solved. Indeed, one encounters pronounced difficulties trying to uniquely identify β and γ only based on the number of newly reported cases for a single outbreak. In such context, there tends to be a deal of information on $R_0 = \beta/\gamma$, which determines the final size of the epidemic and the invasion speed. But relatively little information is available on β and γ individually. From a geometrical point of view, there is a ridge in the parameter space with a sharp curvature in the R_0 direction and a smooth curvature in the complementary directions. The likelihood profiles have been used in order to find the MLE. It turns out that this adequately solved the problem.

Very preliminary results, on rotavirus inference are presented. In order to applied this framework to real data, a single age strata without seasonality without gamma noise have been fitted to the rotavirus data from season 2002. It turns out that it is relatively easy to estimate R_0 from the rotavirus data. Unfortunately, the same problem as described in the simulation study arises. Profiles likelihood have been used to compute the MLE of the model. It turns out that without any previous knowledge of the MLE the required computational efforts are heavy. Then a mixed method have been used. First, the MIF algorithm have been used to find the relationship between β and γ. Then likelihood profiles have been computed at selected places to find the MLE. Based on those investigations, a confidence region's prediction have been computed, showing an insufficient coverage of the rotavirus data. More stochasticity is required.

As gamma noise is a sensible parameter during the optimization procedure, it has been computed separately. The model's predictions using extra stochasticity fits better rotavirus data. But more computations are required to draw strong conclusions. One limitation of this approach is

that very little information can be learn from the data about β and γ separately. Then a possible solution is to use the information from several outbreaks. Due to time constraint, it has not been investigated.

A useful feature of this framework is to better asses model parameters uncertainty and model's prediction uncertainty. Indeed, this method gives access to an estimation of the distribution of the model's parameters. Unfortunately, due to lack of time, this feature was not carried on in this study. The next steps is to fit a more complex model to the rotavirus data. A multiple age strata that can manage seasonality is certainly a good candidate.

Bibliography

Daryl J Daley, Joe Gani, and Joseph Mark Gani. *Epidemic modelling: an introduction*, volume 15. Cambridge University Press, 2001.

John Graunt. *Natural and Political Observations made upon the Bills of Mortality*. Number 2 in 1. The Johns Hopkins Press, 1939.

Fred Brauer, Carlos Castillo-Chavez, and Carlos Castillo-Chavez. *Mathematical models in population biology and epidemiology*, volume 1. Springer, 2001.

Felix Weidemann, Manuel Dehnert, Judith Koch, Ole Wichmann, and Michael Höhle. Modelling the epidemiological impact of rotavirus vaccination in Germany – A Bayesian approach. *Vaccine*, 32(40):5250–5257, 2014a.

McKendrick AG Kermack WO. A contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society*, A 115(772):700–721, August 1927.

Odo Diekmann, JAP Heesterbeek, and Johan AJ Metz. On the definition and the computation of the basic reproduction ratio R0 in models for infectious diseases in heterogeneous populations. *Journal of mathematical biology*, 28(4):365–382, 1990.

Maurice S Bartlett. Measles periodicity and community size. *Journal of the Royal Statistical Society. Series A (General)*, pages 48–70, 1957.

Tom Britton and Federica Giardina. Introduction to statistical inference for infectious diseases. *arXiv preprint arXiv:1411.3138*, 2014.

Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The journal of physical chemistry*, 81(25):2340–2361, 1977.

Priscilla E Greenwood and Luis F Gordillo. Stochastic epidemic modeling. In *Mathematical and Statistical Estimation Approaches in Epidemiology*, pages 31–52. Springer, 2009.

Thomas G Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of applied Probability*, 7(1):49–58, 1970.

Penelope H Dennehy. Transmission of rotavirus and other enteric pathogens in the home. *The Pediatric infectious disease journal*, 19(10):S103–S105, 2000.

Jacqueline E Tate, Anthony H Burton, Cynthia Boschi-Pinto, A Duncan Steele, Jazmin Duque, and Umesh D Parashar. 2008 estimate of worldwide rotavirus-associated mortality in children younger than 5 years before the introduction of universal rotavirus vaccination programmes: a systematic review and meta-analysis. *The Lancet Infectious Diseases*, 12(2):136 – 141, 2012. ISSN 1473-3099. doi: http://dx.doi.org/10.1016/S1473-3099(11)70253-5. URL http://www.sciencedirect.com/science/article/pii/S1473309911702535.

David I Bernstein. Rotavirus overview. *The Pediatric infectious disease journal*, 28(3):S50–S53, 2009.

Judith Koch and Miriam Wiese-Posselt. Epidemiology of rotavirus infections in children less than 5 years of age: Germany, 2001–2008. *The Pediatric infectious disease journal*, 30(2): 112–117, 2011.

Felix Weidemann, Manuel Dehnert, Judith Koch, Ole Wichmann, and Michael Höhle. Bayesian parameter inference for dynamic infectious disease modelling: rotavirus in Germany. *Statistics in medicine*, 33(9):1580–1599, 2014b.

Bettina M Rosner, Klaus Stark, and Dirk Werber. Epidemiology of reported Yersinia enterocolitica infections in Germany, 2001-2008. *BMC Public Health*, 10(1):337, 2010.

Federal Bureau of Statistics. GENESIS Online Database, 2013.

William Atkinson, S Wolfe, and Jennifer Hamborsky. *Epidemiology and prevention of vaccine-preventable diseases*. Public Health Foundation, 2011.

Virginia E Pitzer, Cécile Viboud, Ben A Lopman, Manish M Patel, Umesh D Parashar, and Bryan T Grenfell. Influence of birth rates and transmission rates on the global seasonality of rotavirus incidence. *Journal of the Royal Society Interface*, page rsif20110062, 2011.

National Immunisation Committee et al. Immunisation guidelines for Ireland. *Royal College of Physicians of Ireland*, 1996.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. *An introduction to sequential Monte Carlo methods*. Springer, 2001.

M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.

Jeff A Bilmes et al. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, 4(510):126, 1998.

EL Ionides, C Bretó, and AA King. Inference for nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 103(49):18438–18443, 2006.

William J Anderson. *Continuous-time Markov chains: an applications-oriented approach*. Springer Science and Business Media, 2012.

Erhan Çınlar. Poisson Random Measures. In *Probability and Stochastics*, pages 243–312. Springer, 2011.

Carles Bretó, Daihai He, Edward L Ionides, and Aaron A King. Time series analysis via mechanistic models. *The Annals of Applied Statistics*, pages 319–348, 2009.

Xiaodong Cai and Zhouyi Xu. K-leap method for accelerating stochastic simulation of coupled chemical reactions. *The Journal of chemical physics*, 126(7):074102–074102, 2007.

Olivier Cappé, Eric Moulines, and Tobias Rydén. *Inference in hidden Markov models*. Springer Science & Business Media, 2006.

Mark EJ Newman, Gerard T Barkema, and MEJ Newman. *Monte Carlo methods in statistical physics*, volume 13. Clarendon Press Oxford, 1999.

Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.

José L Sanz-González, Diego Andina, and Juan Seijas. Importance sampling and mean-square error in neural detector training. *Neural processing letters*, 16(3):259–276, 2002.

John Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica: Journal of the Econometric Society*, pages 1317–1339, 1989.

Jun S Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)*, volume 140 (2), pages 107–113. IET, 1993.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL http://www.R-project.org/.

Aaron A King, Dao Nguyen, and Edward L Ionides. Statistical inference for Partially Observed Markov Processes via the R Package pomp. *Journal of Statistical Software*, 2014.

Aaron A. King, Edward L. Ionides, Carles Martinez Bretó, Stephen P. Ellner, Matthew J. Ferrari, Bruce E. Kendall, Michael Lavine, Dao Nguyen, Daniel C. Reuman, Helen Wearing, and Simon N. Wood. *pomp: Statistical Inference for Partially Observed Markov Processes*, 2015a. URL http://kingaa.github.io/pomp. R package, version 1.1.1.1.

Aaron A. King, Dao Nguyen, and Edward L. Ionides. Statistical inference for partially observed Markov processes via the R package pomp. *Journal of Statistical Software*, in press, 2015b.

Edward L. Ionides, Anindya Bhadra, Yves Atchadé, and Aaron King. Iterated filtering. *Ann. Statist.*, 39(3):1776–1802, 06 2011. doi: 10.1214/11-AOS886. URL http://dx.doi.org/10.1214/11-AOS886.

Revolution Analytics and Steve Weston. *foreach: Foreach looping construct for R*, 2014a. URL http://CRAN.R-project.org/package=foreach. R package version 1.4.2.

Revolution Analytics. *doMC: Foreach parallel adaptor for the multicore package*, 2014. URL http://CRAN.R-project.org/package=doMC. R package version 1.3.3.

Revolution Analytics and Steve Weston. *doParallel: Foreach parallel adaptor for the parallel package*, 2014b. URL http://CRAN.R-project.org/package=doParallel. R package version 1.0.8.

Pierre L'ecuyer, Richard Simard, E Jack Chen, and W David Kelton. An object-oriented random-number package with many long streams and substreams. *Operations research*, 50 (6):1073–1075, 2002.

Jun S Liu, Rong Chen, and Tanya Logvinenko. A theoretical framework for sequential importance sampling with resampling. In *Sequential Monte Carlo methods in practice*, pages 225–246. Springer, 2001.

J Spall. Introduction to stochastic searchand optimization: Estimation, simulation and control. *Wiley*, 34:54–58, 2003.

Mark H Holmes. *Introduction to perturbation methods*, volume 20. Springer Science & Business Media, 2012.

List of Figures

List of Tables

7 Annexes

7.1 Chapter 1: Introduction

R code used to produce the plots of chapter 1 are presented below. A Gillespie algorithm for simulating stochastic SIR models and the same code updated for parallel use. The parallel code perform poorly for a single simulation. This is why two codes are presented. Indeed, in Gillespie algorithm the total number of simulation is stochastic thus when aggregating those an augmentation step as to be added in order that all the simulations have the same length in order to be returned in a matrix object.

```
################################################
### SIR simulations Gillespie
################################################
#####################################################################
# Author: Gilles Kratzer <gilles.kratzer@gmail.com>
# Info:
#
# History
# -- 08/04/2015 File created
# -- 21/08/2015 Functionalization/speedup code
#####################################################################
#Paramers:
# parms
# - m: birth and death rate in the compartments
# - b: force of infection
# - v: (additional) death rate due to infection
# - r: recovery rate
# X0: initial state (S, I, R)
# Time.window: (min, max)
# processes: a 7x3 matrix
# - "birth"
# - "death.S"
# - "death.I"
# - "death.R"
# - "infection"
# - "death.infec"
# - "recovery"
# - c("dS","dI","dR")
#
# Returns:
#  a matrix containing :
# Time:time step of the evaluation of the compartments number
# A number of column (size of start) that contain the number of individuals
# in each compartments for each time step
#####################################################################

##package

##seed
set.seed(01042015, kind = "L'Ecuyer-CMRG")

##Function

gillespie<-function(parms=c(m=1e-4,b=0.02,v=0.1,r=0.3) , X0=c(S=97, I=3, R=0), time.window=c(0,100), processes=processes, pb=FALSE){

  # initialize state and time variables and write them into output
  X <- X0
```

```
  time <-  time.window[1]

  # define output dataframe
  out <- data.frame(t=time,S=X0["S"], I=X0["I"], R=X0["R"],row.names=1)


  # process probabilities
  probabilities <- function(X, parms){

    a<-matrix(data = 0,nrow = 7,ncol = 1)
    a[1] = parms["m"]*(X["S"]+X["I"]+X["R"])
    a[2] = parms["m"]*X["S"]
    a[3] = parms["m"]*X["I"]
    a[4] = parms["m"]*X["R"]
    a[5] = parms["b"]*X["S"]*X["I"]
    a[6] = parms["v"]*X["I"]
    a[7] = parms["r"]*X["I"]
    return(a)
}

##Progress bar
if(pb==TRUE) pbPrint <- txtProgressBar(min = 0, max = time.window[2], style = 3)

while(time < time.window[2] & X["I"]>0){

  # calculate process probabilities for current state
  a<-probabilities(parms = parms, X = X)

  # WHEN does the next process happen?
  elapsed.time <- rexp(1, rate=sum(a))

  # update time
  time<-time+elapsed.time

  # Which process happens % update states
  which.trans <-sample(length(a),1,prob=a)
  X<-X+processes[which.trans,]

  # write into output
  out <- rbind(out,c(time,X))

  # update progress bar
  if(pb==TRUE){setTxtProgressBar(pb = pbPrint, value = time)}
}
out
}
```

```
################################################
### SIR simulations Gillespie updated for parallel use
################################################
#######################################################################
# Author: Gilles Kratzer <gilles.kratzer@gmail.com>
# Info:
#
# History
# -- 08/04/2015 File created
# -- 21/08/2015 Functionalization/speedup code
#######################################################################
#Paramers:
# parms
# - m: birth and death rate in the compartments
# - b: force of infection
# - v: (additional) death rate due to infection
# - r: recovery rate
# X0: initial state (S, I, R)
# Time.window: (min, max)
# processes: a 7x3 matrix
# - "birth"
# - "death.S"
# - "death.I"
# - "death.R"
# - "infection"
# - "death.infec"
# - "recovery"
# - c("dS","dI","dR")
#
# Returns:
#  a matrix containing :
# Time: time step of the evaluation of the compartments number
# A number of column (size of start) that contain the number of individuals
# in each compartments for each time step
#######################################################################

##package
library(foreach)
library(iterators)
library(doParallel)
library(snow)
library(plyr)
library(nws)
```

```
##seed
set.seed(01042015, kind = "L'Ecuyer-CMRG")

##Function
gillespieMultTraj<-function(parms=c(m=1e-4,b=0.02,v=0.1,r=0.3) , X0=c(S=97, I=3, R=0), time.window=c(0,100), processes=processes, NbTraj=100, NbCP

  # Start the clock!
  if(ClockTime==TRUE) ptm <- proc.time()

gillespie<-function(parms=c(m=1e-4,b=0.02,v=0.1,r=0.3) , X0=c(S=97, I=3, R=0), time.window=c(0,100),
                    processes=processes, maxLine=1000){

  # initialize state and time variables and write them into output
  X <- X0

  time <-  time.window[1]

  # define output dataframe
  out <- data.frame(t=time,S=X0["S"], I=X0["I"], R=X0["R"],row.names=1)


  # process probabilities
  probabilities <- function(X, parms){

    a<-matrix(data = 0,nrow = 7,ncol = 1)
    a[1] = parms["m"]*(X["S"]+X["I"]+X["R"])
    a[2] = parms["m"]*X["S"]
    a[3] = parms["m"]*X["I"]
    a[4] = parms["m"]*X["R"]
    a[5] = parms["b"]*X["S"]*X["I"]
    a[6] = parms["v"]*X["I"]
    a[7] = parms["r"]*X["I"]
    return(a)
  }


  while(time < time.window[2] & X["I"]>0){

    # calculate process probabilities for current state
    a<-probabilities(parms = parms, X = X)

    # WHEN does the next process happen?
    elapsed.time <- rexp(1, rate=sum(a))

    # update time
    time<-time+elapsed.time

    # Which process happens % update states
    which.trans <-sample(length(a),1,prob=a)
    X<-X+processes[which.trans,]

    # write into output
    out <- rbind(out,c(time,X))

  }
  l<-length(out[,1])
  while(l < maxLine){
    out <- rbind(out,c(NA, NA, NA, NA))
    l=l+1
  }
  return(out)
}

#array of results

gillespie.mult.traj<-array(data = NA, dim = c(maxLine,4, NbTraj))

######################
#Parallel Computations
######################

#setup parallel backend to use CPU efficiently
#cl2<-makeCluster(detectCores(logical = FALSE), "localhost")
registerDoParallel(cl = detectCores(logical = FALSE), cores = detectCores(logical = FALSE))

gillespie.mult.traj<-foreach(icount(NbTraj), .packages="foreach", .combine = "cbind") %dopar% {
  gillespie(parms, X0, time.window, processes, maxLine)
}

#close multicore
#stopCluster(cl2)

gillespie.mult.traj.array<-array(NA, dim = c(dim(gillespie.mult.traj)[1],4, NbTraj))
gillespie.mult.traj.array[,1,]<-data.matrix(gillespie.mult.traj[,c(TRUE,rep(FALSE,3))], rownames.force = NA)
gillespie.mult.traj.array[,2,]<-data.matrix(gillespie.mult.traj[,c(FALSE,TRUE,rep(FALSE,2))], rownames.force = NA)
gillespie.mult.traj.array[,3,]<-data.matrix(gillespie.mult.traj[,c(FALSE, FALSE,TRUE,FALSE)], rownames.force = NA)
gillespie.mult.traj.array[,4,]<-data.matrix(gillespie.mult.traj[,c(FALSE, FALSE,FALSE, TRUE)], rownames.force = NA)
# Stop the clock!
if(ClockTime==TRUE){
  ClockTime<-proc.time() - ptm
  cat("user" , "system", "elapsed", "\n", ClockTime)
}
return(gillespie.mult.traj.array)
}
```

7.2 Chapter 2: Description of the epidemiological data

In the table below is reproduce the estimates of the basic reproduction number issued from Pitzer et al. [2011].

Table 7.1: Basic Reproduction Number for different country

Country	Estimated R_0 (mean = 59.48)	95% CI
Australia	53.9	(52.2, 55.6)
Taiwan	23.3	(20.4, 26.2)
USA	45.3	(44.8, 45.8)
China	88.1	(83.4, 93.4)
Nepal	51.3	(43.2, 61.9)
Uzbekistan	46.7	(42.4, 51.5)
Hong Kong SAR	52.1	(48.7, 55.4)
Cambodia	68.5	(62.0, 76.1)
Fiji	46.2	(36.8, 58.6)
Lao PDR	34.3	(30.4, 38.8)
England and Wales	54.4	(54.0, 54.8)
Bangladesh	72.2	(68.4, 76.4)
Malawi	191	(137, 313)
Nigeria	37.0	(27.5, 51.9)
Philippines	27.9	(24.4, 31.6)

7.3 Chapter 3: Theoretical background

Intuitively, a filtration is all the historical information available at a certain time on a stochastic process. More formally, let us assume a probability space $(\Omega, \mathcal{F}, \mathbf{P})$. A filtration is an increasing sequence of σ-algebras on a measurable space $(\Omega, \mathcal{F})$. A filtration is a sequence of σ-algebras $\{\mathcal{F}_t\}_{t\geq 0}$ with $\mathcal{F}_t \subseteq \mathcal{F}$ for each t such that $t_1 \leq t_2 \implies \mathcal{F}_{t_1} \subseteq \mathcal{F}_{t_2}$. Classically, a σ-algebra defines the set of events that can be measured. A filtration represents the information available up to and including each time t for a stochastic process. In other words, it becomes more and more precise as the set of measurable events is staying the same or increasing. In our case, filtration containing Poisson processes.

A random measure Çınlar [2011] is a measure-valued of the generalization of a random variable. More formally, let (E, ε) be a measurable space and let $(\Omega, \mathcal{F}, \mathbf{P})$ be a probability space. A random measure on (E, ε) is a transition kernel from $(\Omega, \mathcal{F})$ into (E, ε). A mapping $M : \Omega\ x\ \varepsilon \to \mathbb{R}_+$ is called a random measure if $\omega \to M(\omega, A)$ is a random variable $\forall\ A \subset E$ and if $A \to M(\omega, A)$ is a measure on $(E, \varepsilon) \forall \omega \in \Omega$. A Poisson random measure is defined in using a measurable space (E, ε) and let ν be a measure on it. A random measure N on (E, ε) is said to be Poisson with mean ν if $\forall A \in \varepsilon$, the random variable N(A) has a Poisson distribution with mean $\nu(A)$ and whenever $A_1, ..., A_n$ are in ε and disjoint, the random variables $N(A_1), ..., N(A_n)$ are independent $\forall n \geq 2$.

R code used to produce the plots of chapter 3 are presented below. The mechanistic model based on Euler multinomial method with gamma noise described in Bretó et al. [2009] and the same code updated for parallel use are presented. The parallel code perform poorly for a single simulation. This is why two codes are presented.

```
######################################################################
### SIR mechanistic model
######################################################################
# Author: Gilles Kratzer <gilles.kratzer@gmail.com>
# Info: Euler scheme for a numerical solution of the MC using gamma noise
#       Based on paper "Time series analysis via mechanistic models"
#
# History:
# -- 23/03/2015 file created
# -- 01/04/2015 files modified for presentation
# -- 08/04/2015 beginning of versioning
# -- 16/04/2015 optimization based on MH comments
# -- 17/04/2015 progress bar
######################################################################
#Paramers:
# T - The system is simulated for the time period [0,T]
# N - Number of
# start - Start status of the compartments (X[t=0])
# delta - scale of gamma noise
# sigma - shape of the gamma noise
# mu - flux in between compartments
# pb - TRUE/FALSE progress bar display during computation
#
# Returns:
#  a matrix containing :
# Time: time step of the evaluation of the compartments number
# A number of column (size of start) that contain the number of individuals
# in each compartments for each time step
######################################################################

##package

##seed
set.seed(01042015, kind = "L'Ecuyer-CMRG")

euler<-function(T=100, N=10, start, delta=0.1 ,sigma=0.1, mu=0.1, pb=FALSE){

  ##constructing times
  delta.time<-T/N
  times<-matrix(data = seq(from = 0,by = delta.time,length.out = N), nrow = 1, ncol = N)

  ##Initialisation of compartments
  nbCompartments<-length(start)

  ##initialization of the output
  out<-matrix(nrow = length(times), ncol = nbCompartments+1)
  out[1,]<-c(0,start)

  ##sigma to matrix
  if(is.matrix(sigma)) sigma.mat<-sigma else sigma.mat<-matrix(data = sigma,nrow = nbCompartments, ncol = nbCompartments)

  ##Progress bar
  if(pb==TRUE) pbPrint <- txtProgressBar(min = 0, max = length(times[1,]), style = 3)

  ##loop over time
  for(i in seq_len(length(times[1,])-1)){

    # update progress bar
    if(pb==TRUE){setTxtProgressBar(pbPrint, i)}

    ##Generating noise increments
    gam<-apply(X = sigma.mat,MARGIN = c(1,2),FUN = function(x) rgamma(n = 1,scale =x^2 ,shape = delta/x^2))

    ##generate probabilities & increments
    #initialization
    proba<-matrix(nrow = nbCompartments, ncol = nbCompartments)
    var.N<-matrix(nrow = nbCompartments, ncol = nbCompartments)

    ##Probabilities
    proba<-(1-exp(-rowSums(mu * gam)))*(mu * gam)/(rowSums(mu * gam))

    ##Normalization
    diag(proba) <- 1 - rowSums(proba)

    ##generate process increments
    ###hoehle: I guess this can't be vectorized s.t. rmultinom(n=nCompartments, size=X[i,], proba) (I don't think so buch check)
    for(c in 1:nbCompartments){
      var.N[c,]<-rmultinom(n=1, size = out[i,c+1], prob = proba[c,])
    }
  out[i+1,-1]<-colSums(var.N)
  }
  #Output: time and compartments number at each time (# of compartments +1)
  out[,1] <- times
  return(out)
  #closing pb
  close(pb)
```

```
}
```

```
#####################################################################
### SIR mechanistic model
#####################################################################
# Author: Gilles Kratzer <gilles.kratzer@gmail.com>
# Info: Euler scheme for a numerical solution of the MC using gamma noise
#       Based on paper "Time series analysis via mechanistic models"
#       Multiple trajectories implemented in parallel
#       Optimized for 2 CPU
#
# History:
# -- 23/03/2015 file created
# -- 01/04/2015 files modified for presentation
# -- 08/04/2015 beginning of versioning
# -- 16/04/2015 optimization based on MH comments
# -- 17/04/2015 progress bar
# -- 24/04/2015 multiple trajectories in parallel
#####################################################################
#Paramers:
# T - The system is simulated for the time period [0,T]
# N - Number of
# start - Start status of the compartments (X[t=0])
# delta - scale of gamma noise
# sigma - shape of the gamma noise
# mu - flux in between compartments
# pb - TRUE/FALSE progress bar display during computation
#
# Returns:
#  a matrix containing :
# Time: time step of the evaluation of the compartments number
# A number of column (size of start) that contain the number of individuals
# in each compartments for each time step
#####################################################################

##package
library(foreach)
library(utils)
library(iterators)
library(doParallel)
library(snow)

##seed
set.seed(01042015, kind = "L'Ecuyer-CMRG")

EulerMultTraj<-function(T=100, N=10, start, delta=0.1 ,sigma=0.1, mu=0.1, NbTraj=100, NbCPU=2, ClockTime=FALSE)
{
  # Start the clock!
  if(ClockTime==TRUE) ptm <- proc.time()

  ##Core function optimized for parallel computing
  euler<-function(T=100, N=10, start, delta=0.1 ,sigma=0.1, mu=0.1){

    ##constructing times
    delta.time<-T/N
    times<-matrix(data = seq(from = 0,by = delta.time,length.out = N), nrow = 1, ncol = N)

    ##Initialisation of compartments
    nbCompartments<-length(start)

    ##initialization of the output
    out<-matrix(nrow = length(times), ncol = nbCompartments+1)
    out[1,]<-c(0,start)

    ##sigma to matrix
    if(is.matrix(sigma)) sigma.mat<-sigma else sigma.mat<-matrix(data = sigma,nrow = nbCompartments, ncol = nbCompartments)

    ##loop over time
    for(i in seq_len(length(times[1,])-1)){

      ##Generating noise increments
      gam<-apply(X = sigma.mat,MARGIN = c(1,2),FUN = function(x) rgamma(n = 1,scale =x^2 ,shape = delta/x^2))

      ##generate probabilities & increments
      #initialization
      proba<-matrix(nrow = nbCompartments, ncol = nbCompartments)
      var.N<-matrix(nrow = nbCompartments, ncol = nbCompartments)

      ##Probabilities
      proba<-(1-exp(-rowSums(mu * gam)))*(mu * gam)/(rowSums(mu * gam))

      ##Normalization
      diag(proba) <- 1 - rowSums(proba)

      ##generate process increments
      for(c in 1:nbCompartments){
        var.N[c,]<-rmultinom(n=1, size = out[i,c+1], prob = proba[c,])
      }
      out[i+1,-1]<-colSums(var.N)
    }
    #Output: time and compartments number at each time (# of compartments +1)
    out[,1] <- times
    return(out)
```

```
    }

#array of results
Euler.mult.traj<-array(dim = c(N,4, NbTraj))

#####################
#Parallel Computation#
#####################

#setup parallel backend to use CPU efficiently
registerDoParallel(cl = detectCores(logical = FALSE), cores = detectCores(logical = FALSE))

Euler.mult.traj<-foreach(i = icount(NbTraj), .packages="foreach", .combine = "cbind") %dopar% {
  euler(T, N, start, delta,sigma, mu)
}


Euler.mult.traj.array<-array(dim = c(N,4, NbTraj))
Euler.mult.traj.array[,1,]<-Euler.mult.traj[,c(TRUE,rep(FALSE,3))]
Euler.mult.traj.array[,2,]<-Euler.mult.traj[,c(FALSE,TRUE,rep(FALSE,2))]
Euler.mult.traj.array[,3,]<-Euler.mult.traj[,c(FALSE, FALSE,TRUE,FALSE)]
Euler.mult.traj.array[,4,]<-Euler.mult.traj[,c(FALSE, FALSE,FALSE, TRUE)]
# Stop the clock!
if(ClockTime==TRUE) ClockTime<-proc.time() - ptm

cat("user" , "system", "elapsed", "\n", ClockTime)
return(Euler.mult.traj.array)
}
```

7.3.1 Theorem of the Maximum Likelihood Estimation via Iterated Filtering

Before introducing the theorem 7.3.1, let us recall some of the Landau notation Holmes [2012]. **Big O notation** written $\mathcal{O}$ is defined for two real functions f and g if and only if there is a positive constant M such that for all sufficiently large values of x one can write $\mid f(x) \mid \leq M \mid g(x) \mid, \forall x > x_0$ then one can write $f(x) = \mathcal{O}(g(x))$ as $x \to \infty$. In addition, **Small o notation** is defined for two real functions f and g if and only if for all sufficiently large values of x one can write $\mid f(x) \mid \leq M \mid g(x) \mid, \forall x > x_0$ and $\forall M$ then one can write $f(x) = o(g(x))$ as $x \to \infty$.

Theorem 7.3.1. *Assuming that:*

- *The Hessian matrix is bounded i.e* $\exists B > 0$ *and* $\sigma_0 > 0$ *such that,* $\forall \sigma < \sigma_0$ *and all* $\theta_t \in \mathbb{R}^{d\theta}$*, where* d_θ *is the dimension of the parameter space* $\mid \nabla^2 f_t(\theta_t, \sigma) \mid < B$ *(7.1)*
- $\mathbf{E}[\mid \theta_t - \hat{\theta}_{t-1} \mid^2 \mid y_{1:t-1}] = \mathcal{O}(\sigma^2)$ *(7.2)*
- $\mathbf{E}[\mid \theta_t - \hat{\theta}_{t-1} \mid^3 \mid y_{1:t-1}] = o(\sigma^2)$ *(7.3)*

Then:

$$\lim_{\sigma \to 0} \sum_{t=1}^{T} V_t^{-1}(\hat{\theta}_t - \hat{\theta}_{t-1}) = \nabla \log f(y_{1:T} \mid \theta, \sigma = 0), \tag{7.4}$$

where $[\nabla f(x)]_i = \frac{\delta f}{\delta x_i}$ *and* $\hat{\theta}_0 = \theta$*. Furthermore, for a sequence* $\sigma_n \to 0$ *define* $\hat{\theta}^{(n)}$ *recursively defined by :*

$$\hat{\theta}^{(n+1)} = \hat{\theta}^{(n)} + V_{1,n} \sum_{t=1}^{T} V_{t,n}^{-1}(\hat{\theta}_t^{(n)} - \hat{\theta}_{t-1}^{(n)}) \tag{7.5}$$

where $\hat{\theta}_t^{(n)} = \hat{\theta}_t(\hat{\theta}^{(n)}, \sigma_n)$ *and* $V_{t,n} = V_t(\hat{\theta}^{(n)}, \sigma_n)$*. If there exist a* $\hat{\theta}$ *with* $\mid \hat{\theta}^{(n)} - \hat{\theta} \mid / \sigma_n^2 \to 0$ *then* $\nabla \log f(y_{1:T} \mid \theta = \hat{\theta}, \sigma = 0) = 0$

Some remarks and further computations are welcome in order to understand the meaning of the Theorem 7.3.1. It asserts that for sufficiently small σ the algorithm 2 (MIF) will update the parameter estimate such that the likelihood increases. Indeed equation (7.4) shows that, in the limit $\sigma \to 0$, this equation tends to zero. As the gradient of the log likelihood is equal to zero only for the maximum likelihood. Thus equation (7.5) simplifies to $\hat{\theta}^{(n+1)} = \hat{\theta}^{(n)}$. And then give a stable estimation for $\hat{\theta}$.

$$\begin{aligned}
\hat{\theta}^{(n+1)} &= \hat{\theta}^{(n)} + V_{1,n} \sum_{t=1}^{T} V_{t,n}^{-1} (\hat{\theta}_t^{(n)} - \hat{\theta}_{t-1}^{(n)}) \\
&= \hat{\theta}^{(n)} + (\hat{\theta}_1^{(n)} - \hat{\theta}_0^{(n)}) + \frac{V_{1,n}}{V_{2,n}} (\hat{\theta}_2^{(n)} - \hat{\theta}_1^{(n)}) + \cdots + \frac{V_{1,n}}{V_{T,n}} (\hat{\theta}_T^{(n)} - \hat{\theta}_{T-1}^{(n)}) \\
&= \hat{\theta}_1^{(n)} \left(\frac{V_{1,n}}{V_{1,n}} - \frac{V_{1,n}}{V_{2,n}} \right) + \hat{\theta}_2^{(n)} \left(\frac{V_{1,n}}{V_{2,n}} - \frac{V_{1,n}}{V_{3,n}} \right) + \cdots + \hat{\theta}_T^{(n)} \left(\frac{V_{1,n}}{V_{T-1,n}} - \frac{V_{1,n}}{V_{T,n}} \right) \\
&= V_{1,n} \left\{ \sum_{t=1}^{T-1} (V_{t,n}^{-1} - V_{t+1,n}^{-1}) \hat{\theta}_t^{(n)} + V_{T,n}^{-1} \hat{\theta}_T^{(n)} \right\}
\end{aligned} \tag{7.6}$$

In equation (7.6), one uses the following equality $\hat{\theta}_0^{(n)} = \hat{\theta}^{(n)}$. Following equation (7.6), equation (7.7) shows that the estimate $\hat{\theta}^{(n+1)}$ can be viewed as a matrix weighted average of the previous estimates depending on time $\hat{\theta}_{1:T}^{(n)}$.

$$\begin{aligned}
V_{1,n} \left\{ \sum_{t=1}^{T-1} (V_{t,n}^{-1} - V_{t+1,n}^{-1}) + V_{T,n}^{-1} \right\} &= \frac{V_{1,n}}{V_{1,n}} - \frac{V_{1,n}}{V_{2,n}} + \frac{V_{1,n}}{V_{2,n}} - \frac{V_{1,n}}{V_{3,n}} + \cdots + \frac{V_{1,n}}{V_{T-1,n}} - \frac{V_{1,n}}{V_{T,n}} + \frac{V_{1,n}}{V_{T,n}^{-1}} \\
&= \frac{V_{1,n}}{V_{1,n}} = \mathbb{1}_{d_\theta}
\end{aligned} \tag{7.7}$$

Where $\mathbb{1}_{d_\theta}$ is the identity matrix of size d_θ x d_θ. Let us introduce the following notation: $f_t(\psi) = f(y_t \mid y_{1:t-1}, \theta_t = \psi)$.

Proof. Suppose inductively that $\mid V_t \mid = \mathcal{O}(\sigma^2)$ and $\mid \hat{\theta}_{t-1} - \theta \mid = \mathcal{O}(\sigma^2)$. This holds for $t = 1$ by construction. Bayes' formula gives :

$$\begin{aligned}
\frac{f(\theta_t \mid y_{1:t})}{f(\theta_t \mid y_{1:t-1})} &= \frac{f(\theta_t, y_{1:t-1}, y_t)}{f(\theta_t, y_{1:t-1})} \frac{f(y_{1:t-1})}{f(y_{1:t})} \\
&= \frac{f(y_t \mid \theta_t, y_{1:t-1}) f(\theta_t, y_{1:t-1}) f(y_{1:t-1})}{f(\theta_t, y_{1:t-1}) f(y_{1:t})} \\
&= \frac{f_t(\theta_t)}{f(y_t \mid y_{t-1})} \\
&= \frac{f_t(\theta_t)}{\int f(y_t \mid y1 : t-1, \theta_t) f(\theta_t \mid y1 : t-1) d\theta_t} \\
&= \frac{f_t(\hat{\theta}_{t-1}) + (\theta_t - \hat{\theta}_{t-1}) \nabla f_t(\hat{\theta}_{t-1}) + R_t}{f_t(\hat{\theta}_{t-1}) + \mathcal{O}(\sigma^2)} \\
&= \left\{ 1 + (\theta_t - \hat{\theta}_{t-1}) \nabla \log f_t(\hat{\theta}_{t-1}) + \frac{R_t}{f_t(\hat{\theta}_{t-1})} \right\} \times (1 + \mathcal{O}(\sigma^2))
\end{aligned} \tag{7.8}$$

Where R_t is the rest of the Taylor expansion. Equations (7.1) and (7.3) imply $\mid R_t \mid < B \mid \frac{(\theta_t - \hat{\theta}_{t-1})^2}{2} \mid$. In equation (7.8) a Taylor expansion of $f_t(\hat{\theta}_t) = f(y_t \mid y_{1:t-1}, \hat{\theta}_t)$ is used. We further compute :

$$\begin{aligned}
(\hat{\theta}_t - \hat{\theta}_{t-1}) &= \mathbf{E}[\theta_t \mid y_{1:t}] - \hat{\theta}_{t-1} \\
&= \mathbf{E}[\theta_t - \hat{\theta}_{t-1} \mid y_{1:t}] \\
&= \int (\hat{\theta}_t - \hat{\theta}_{t-1}) f(\theta_t \mid y_{1:t}) d\theta_t \\
&= \int (\hat{\theta}_t - \hat{\theta}_{t-1}) \frac{f(\theta_t \mid y_{1:t})}{f(\theta_t \mid y_{1:t-1})} f(\theta_t \mid y_{1:t-1}) d\theta_t \\
&= \int (\hat{\theta}_t - \hat{\theta}_{t-1}) \left\{ 1 + (\theta_t - \hat{\theta}_{t-1}) \nabla \log f_t(\hat{\theta}_{t-1}) + \frac{R_t}{f_t(\hat{\theta}_{t-1})} \right\} \text{ x } (1 + \mathcal{O}(\sigma^2)) f(\theta_t \mid y_{1:t-1}) d\theta_t \\
&= \int (\hat{\theta}_t - \hat{\theta}_{t-1})(1 + \mathcal{O}(\sigma^2)) f(\theta_t \mid y_{1:t-1}) d\theta_t \\
&+ \int (\hat{\theta}_t - \hat{\theta}_{t-1})^2 \nabla \log f_t(\hat{\theta}_{t-1})(1 + \mathcal{O}(\sigma^2)) f(\theta_t \mid y_{1:t-1}) d\theta_t \\
&+ \int (\hat{\theta}_t - \hat{\theta}_{t-1}) \frac{R_t}{f_t(\hat{\theta}_{t-1})} (1 + \mathcal{O}(\sigma^2)) f(\theta_t \mid y_{1:t-1}) d\theta_t \\
&= V_t \nabla \log f_t(\hat{\theta}_{t-1}) + o(\sigma^2)
\end{aligned} \tag{7.9}$$

In equation (7.9) the following identity are used $\int (\hat{\theta}_t - \hat{\theta}_{t-1}) f(\theta_t \mid y_{1:t-1}) d\theta_t = 0$ and $\int (\hat{\theta}_t - \hat{\theta}_{t-1})^2 f(\theta_t \mid y_{1:t-1}) d\theta_t = V_t$ and $\int (\hat{\theta}_t - \hat{\theta}_{t-1})^3 f(\theta_t \mid y_{1:t-1}) d\theta_t = o(\sigma)$. Then one can use the inductive argument on equation (7.9) to get the equation (7.10). More precisely, $\mid V_t \mid$ and $\mid \hat{\theta}_{t-1} - \theta \mid$ are $\mathcal{O}(\sigma^2)$ thus :

$$(\hat{\theta}_t - \hat{\theta}_{t-1}) = V_t \nabla \log f_t(\hat{\theta}_{t-1}) + o(\sigma^2) = V_t \nabla \log f_t(\theta, \sigma = 0) + o(\sigma^2) \tag{7.10}$$

A similar argument gives :

$$V_{t+1} = V_t + \sigma^2 \Sigma + o(\sigma^2)$$

Summing equation (7.10) over t gives:

$$\sum_{t=1}^{T} V_t (\hat{\theta}_t - \hat{\theta}_{t-1}) = \sum_{t=1}^{T} \nabla \log f_t(\theta, \sigma = 0) + o(1)$$

And then taking the limit give the required result :

$$
\begin{aligned}
\lim_{\sigma \to 0} \sum_{t=1}^{T} V_t(\hat{\theta}_t - \hat{\theta}_{t-1}) &= \lim_{\sigma \to 0} \sum_{t=1}^{T} \nabla \log f_t(\theta, \sigma = 0) + o(1) \\
&= \sum_{t=1}^{T} \nabla \log f_t(\theta, \sigma = 0) \\
&= \sum_{t=1}^{T} \nabla \log f(y_t \mid y_{1:t-1}, \theta_t = \theta, \sigma = 0) \\
&= \nabla \log \prod_{t=1}^{T} f(y_t \mid y_{1:t-1}, \theta_t = \theta, \sigma = 0) \\
&= \nabla \log f(y_{1:T} \mid \theta, \sigma = 0)
\end{aligned}
\tag{7.11}
$$

Equation (7.11) prove the first part of the theorem 7.3.1.

Second part of the theorem 7.3.1 is proved in using the requirement that $\frac{|\hat{\theta}^{(n)} - \hat{\theta}|}{\sigma_n^2} \to 0$ and continuity argument.

□

7.4 Chapter 5: Results

7.4.1 Model fitting using least square

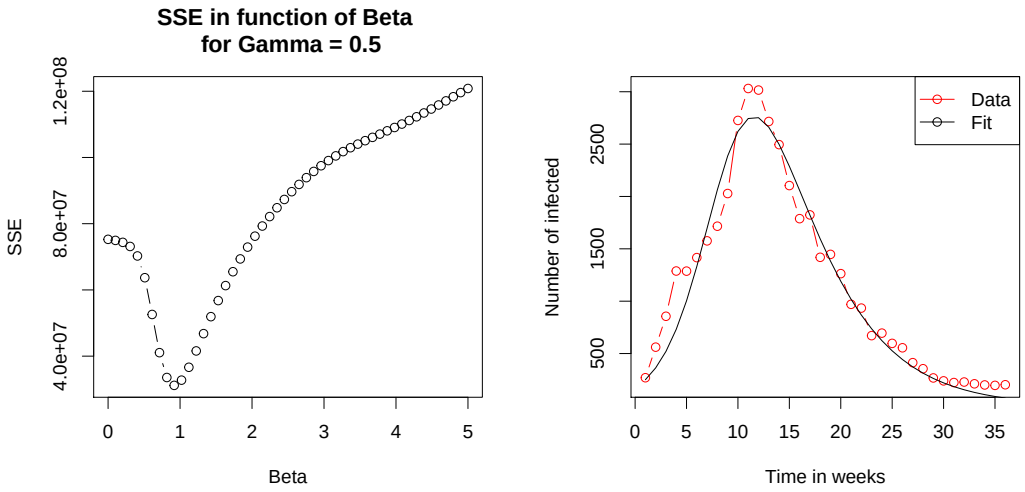

Figure 7.1: Least square estimation of the number of infected

This section is only for learning purpose. The idea is to fit a deterministic single age strata SIR model (without any emission process nor death of birth rate) in order to exemplify the process of parameter inference using a simpler method. This model depends only on two parameters β and γ (see model definition 4.1). Rotavirus data first peak will be fitted, thus from January 2001 - September 2001, thus 37 weeks of data (see figure 2.3). Figure 7.1 presents the SSE (Sum of least Square Estimate) in function β (the force of infection) and the fitting of the data. As one can see the likelihood space is very sensitive to the parameter, even in the simple case. Thus finding the parameter range is a non-trivial task. Fitting procedure give the MLE at $(\beta = 0.63; \gamma = 0.23)$ for initial values $(S(0) = 10000, I(0) = 250, R(0) = 0)$. But the likelihood space slice for $\gamma = 0.5$ change completely the estimate for β. As we will see, it will be an issue in the next section.

Figure 7.2 shows the model produced by the fitting of the data. The idea to use a simple deterministic SIR model to get a rough estimate of the MLE fail for several reason. Mainly, the simple model is too simple, indeed at least an emission process have to be considered.

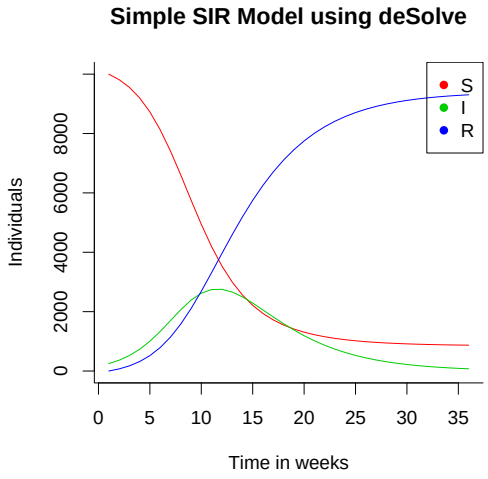

Figure 7.2: Least square estimation of the number of infected

7.4.2 Model implementations

The model implementation using `pomp` package is presented below. Two examples are presented: a single age strata SIR model and a single age strata SIR model including seasonality.

```
##Library
library(pomp)
library(foreach)
library(doMC)
library(parallel)
library(doParallel)

##Options
options(verbose=TRUE)
nbcores<-registerDoMC(detectCores())
nbcores<-registerDoParallel(detectCores())
mcopts <- list(preschedule=FALSE,set.seed=TRUE)

##Seed
set.seed(396658101,kind="L'Ecuyer")

##Data
load(file = "../../Data/data_child.Rdata")

##Model definition

rmeas <- "
cases = rnbinom_mu(theta, rho * H);
"
dmeas <- "
lik = dnbinom_mu(cases, theta, rho * H,give_log);
"

sir.step <- "
double rate[6];
double dN[6];
double P;
P=S+I+R;
rate[0] = mu * P;
rate[1] = beta * I/P;
rate[2] = mu;
rate[3] = gamma;
rate[4] = mu;
rate[5] = mu;
dN[0] = rpois(rate[0] * dt);
reulermultinom(2, S, &rate[1], dt, &dN[1]);

reulermultinom(2, I, &rate[3], dt, &dN[3]);
reulermultinom(1, R, &rate[5], dt, &dN[5]);

S += dN[0] - dN[1] - dN[2];
I += dN[1] - dN[3] - dN[4];
R += dN[3] - dN[5];
H += dN[1];
"
```

```
init <- "
S = popsize-1000;
I = 1000;
R = 0;
H = 0;
"

sir.skel <- '
double rate[6]; // transition rates
double P;

// compute the transition rates
P=S+I+R;
rate[0] = mu * P;
rate[1] = beta * I/P;
rate[2] = mu;
rate[3] = gamma;
rate[4] = mu;
rate[5] = mu;

// assemble the differential equations
DS = rate[0]-rate[1]*S-rate[2]*S;
DI = rate[1]*S-rate[3]*I-rate[4]*I;
DR = rate[3]*I-rate[5]*R;
'

fromEstimationScale <- "
Tgamma = exp(gamma);
Ttheta = exp(theta);
Tbeta = exp(beta);
Tmu = exp(mu);
Trho = expit(rho);
"

toEstimationScale <- "
Tgamma = log(gamma);
Tbeta = log(beta);
Ttheta = log(theta);
Tmu = log(mu);
Trho = logit(rho);
"

##Pomp constructor
sir1<-pomp(data = data.frame(cases = child[1:37], time = seq(0, 1, by=1/36)),
           times = "time", t0 = -1/36,
           dmeasure = Csnippet(dmeas),
           rmeasure = Csnippet(rmeas),
           rprocess = euler.sim(step.fun = Csnippet(sir.step), delta.t = 1/36/20),
           skeleton=Csnippet(sir.skel),
           skeleton.type='vectorfield',
           statenames = c("S", "I", "R", "H"),
           paramnames = c("gamma", "mu", "theta", "beta", "popsize",
                          "rho"), zeronames=c("H"),
           fromEstimationScale = Csnippet(fromEstimationScale),
           toEstimationScale = Csnippet(toEstimationScale),
           initializer = Csnippet(init),
           params = c(popsize = 1000000, beta = 25, gamma = 10,
                      mu = 0.01, rho = 0.15, theta = 1)
)
```

```
##Library
library(pomp)
library(foreach)
library(doMC)
library(parallel)
library(doParallel)
library(rje)

##Options
options(verbose=TRUE)
nbcores<-registerDoMC(detectCores())
nbcores<-registerDoParallel(detectCores())
mcopts <- list(preschedule=FALSE,set.seed=TRUE)

##Seed
set.seed(396658101,kind="L'Ecuyer")

##Data
load(file = "../../Data/data_child.Rdata")


## Warning in readChar(con, 5L, useBytes = TRUE): cannot open compressed file '../../Data/data_child.Rdata', probable reason 'No
such file or directory'
## Error in readChar(con, 5L, useBytes = TRUE): cannot open the connection

##Seasonality: splines

tbasis <- seq(1/52,9,by=1/52)
basis <- periodic.bspline.basis(tbasis,nbasis=3,degree=2,period=10, names="seas%d")

##Model definition
```

```
rmeas <- "
cases = rnbinom_mu(theta, rho * H);
"

dmeas <- "
lik = dnbinom_mu(cases, theta, rho * H, give_log);
"

sir.step <- "
double beta;
double rate[6];
double dN[6];
double P;
double dW; // white noise increment
int k;

P=S+I+R;

// seasonality in transmission
for (k = 0, beta = 0; k < nbasis; k++)
beta += (&beta1)[k]*(&seas1)[k];


// compute the stochasticity
dW = rgammawn(sigma,dt);


rate[0] = mu * P;
rate[1] = (beta*I)/P*(dW / dt);
rate[2] = mu;
rate[3] = (gamma*dW)/dt;
rate[4] = mu;
rate[5] = mu;
dN[0] = rpois(rate[0] * dt); // birth process are poisson

reulermultinom(2, S, &rate[1], dt, &dN[1]);

reulermultinom(2, I, &rate[3], dt, &dN[3]);

reulermultinom(1, R, &rate[5], dt, &dN[5]);

S += dN[0] - dN[1] - dN[2];
I += dN[1] - dN[3] - dN[4];
R += dN[3] - dN[5];
H += dN[1];
"

init <- "
S = (0.2*popsize)-1000;
I = 1000;
R = (0.8*popsize);
H = 0;
"

sir.skel <- '
double rate[6]; // transition rates
//double dW; // white noise increment
int k;
//double seas1;
double beta;
//double beta1;

for (k = 0, beta = 0.0; k < nbasis; k++)
beta += (&beta1)[k]*(&seas1)[k];

// compute the stochasticity
//dW = rgammawn(sigma,dt);


// compute the transition rates
rate[0] = mu*P;
rate[1] = beta*I/P*dW/dt;
rate[2] = mu;
rate[3] = gamma*dW/dt;
rate[4] = mu;
rate[5] = mu;

// assemble the differential equations
DS = rate[0]-rate[1]*S-rate[2]*S;
DI = rate[1]*S-rate[3]*I-rate[4]*I;
DR = rate[3]*I-rate[5]*R;
'

sir.seas<-pomp(
  data = data.frame(cases = child, time = seq(1/52, 9, by=1/52)),
  times = "time", t0 = 1/52,
  dmeasure = Csnippet(dmeas),
  rmeasure = Csnippet(rmeas),
  rprocess = euler.sim(step.fun = Csnippet(sir.step), delta.t = 1/52/20),
  covar=basis,
  tcovar=tbasis,
  #skeleton=Csnippet(sir.skel),
  skeleton.type='vectorfield',
  statenames = c("S", "I", "R", "H"),
  globals="int nbasis = 3;\n",
```

```
  paramnames = c("gamma", "mu", "theta", "beta1", "popsize",
                 "rho", "sigma"), zeronames=c("H"),
  initializer = Csnippet(init),
  params = c(beta=50, beta1=80, beta2=100,gamma=1,mu=1,popsize=1000000, theta=5, rho=0.5, sigma=0.01)
)


## model codes written to '/var/folders/jq/5t7wz8257z91prwp9tds_v080000gn/T//RtmpgZDLLk/pomp_54aeacc8cdcc1b313fc6b171b835f3a0.c'compiling '/var/fo
## link to shared-object library '/var/folders/jq/5t7wz8257z91prwp9tds_v080000gn/T//RtmpgZDLLk/pomp_54aeacc8cdcc1b313fc6b171b835f3a0.so'
```

7.4.3 Model predictions using quantile regression

Figure 7.3 shows 10'000 model's simulations using the MLE and gamma noise. As one can the coverage of the reported 2002 rotavirus data is better than without extra noise. The confidence bands have been computed using quantile regression.

```
## model codes written to '/var/folders/jq/5t7wz8257z91prwp9tds_v080000gn
## link to shared-object library '/var/folders/jq/5t7wz8257z91prwp9tds_v0
```

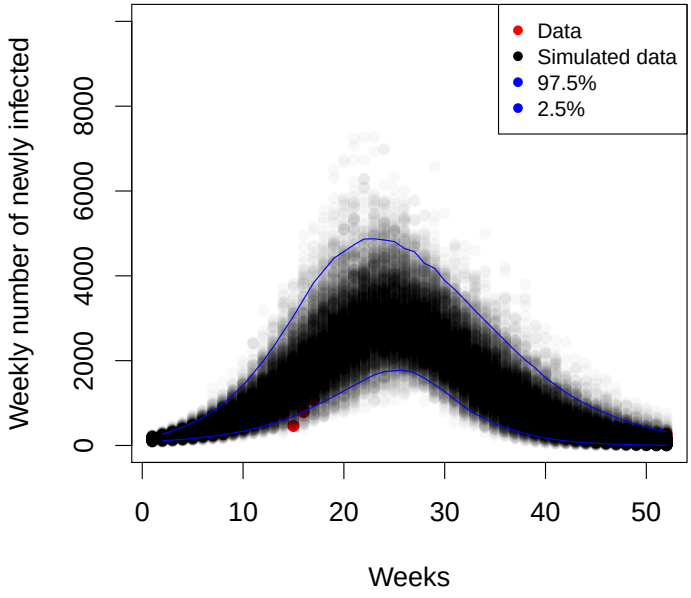

Figure 7.3: Simulations predictions for a complete model