FAST ESTIMATION OF TIME-VARYING TRANSMISSION RATES

FAST ESTIMATION OF TIME-VARYING TRANSMISSION RATES FOR INFECTIOUS DISEASES

By MICHELLE DEJONGE, B.MATH

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Science

Master of Science (2014)
(Mathematics & Statistics)

McMaster University
Hamilton, Ontario

TITLE:	Fast estimation of time-varying transmission rates for infectious diseases
AUTHOR:	Michelle deJonge B.Math (University of Waterloo)
SUPERVISOR:	Dr. David J.D. Earn
NUMBER OF PAGES:	vii, 38 (Supplementary Material: p.40–130)

Abstract

Modelling and analysis of recurrent infectious disease epidemics often depends on the reconstruction of a time-varying transmission rate from historical reports of cases or deaths. Statistically rigorous estimation methods for time-varying transmission rates exist but are too computationally demanding to apply to a time series longer than a few decades. We present a computationally efficient estimation method that is suitable for very long data sets. Our method, which uses a discrete-time approximation to the SIR model for infectious diseases, is easy to implement and outperforms the classic Fine and Clarkson [21] estimation method.

Acknowledgements

First, I would like to sincerely thank my supervisor Professor David Earn for his exemplary guidance over the course of this degree, and for providing me with the opportunity to pursue my research in this field. Thank you for sharing your knowledge and insight, for your generous help, and for the extensive discussions and advice on my research. Thank you to Dr.Benjamin Bolker and Dr.Jonathan Dushoff for being willing to be on my examining committee. Many thanks also to Dora Rosati and Irena Papst for their helpful comments when proofreading my work.

I am so grateful for my family and friends who have constantly surrounded me with love, joy and encouragement. Thank you to my parents for their unwavering support, love, and advice over the years and for encouraging me to pursue my education. A special thank you to my husband Kevin for his continual love, patience, support and invaluable encouragement throughout this degree.

All glory be to Christ.

Contents

1 Introduction and motivation **1**

2 Three methods for estimating the time-varying transmission rate of an infectious disease **2**

2.1 The S method 3

2.2 The S^+ method 5

2.3 The SI method 8

3 Comparing the performance of the three estimation methods **11**

3.1 Simulating case notification data 11

3.1.1 A sinusoidally forced transmission rate 12

3.1.2 Initial conditions 13

3.2 Computing the error in estimation 13

4 Results **15**

4.1 Dependence on parameter values 15

4.2 Sensitivity to incorrect parameter values 18

4.3 Sensitivity to observation error 22

4.4 Sensitivity to process error 24

5 Discussion and conclusions **30**

List of Figures

1 An example of estimating the transmission rate $\beta(t)$ using the S, S^+ and SI methods 16

2 Accuracy in estimating the transmission rate for a variety of parameter values 19

3 Dependence of estimation accuracy on the mean and amplitude of a smooth sinusoidally forced transmission rate 20

4 Dependence of estimation accuracy on the mean and amplitude of a noisy sinusoidally forced transmission rate 21

5 Sensitivity to incorrect parameter values when estimating the transmission rate 23

6 Sensitivity to observation error in estimating the transmission rate 25

7 Estimating $\beta(t)$ in the presence of observation error for a range of reporting and case fatality ratios 26

8 Sensitivity to process error in estimating the transmission rate for a population of $100,000$ 29

9 Sensitivity to process error in estimating the transmission rate for a population of $1,000,000$ 29

List of Tables

1 Notation used in estimating the transmission rate 4

2 Parameter values for measles and smallpox 14

Declaration

I hereby declare that to the best of my knowledge the work presented in this thesis is original and my own work and effort under the supervision of Dr. D.J.D. Earn. Where other sources of information have been used, they have been acknowledged. The material in this thesis is submitted for a Master's degree at McMaster University only and has not been submitted to any other universities.

1 Introduction and motivation

The transmission of an infectious disease throughout a population, is an aspect of disease dynamics that is generally unobservable. Yet without knowledge of the transmission rate, it is difficult to determine which underlying factors (*e.g.,* weather or contact patterns) are most influential in the spread of disease. Mechanistic mathematical modelling provides a lens through which we can discover and interpret information about the spread of a disease, based on quantities we can observe (such as numbers of cases or deaths). Specifically, it provides us with a framework to investigate how different factors affect disease dynamics, and can explain the underlying mechanisms that cause these dynamics to change [2, 4, 7–12, 15, 18, 27, 31, 44].

The transmission rate is a key quantity that influences disease dynamics in epidemic models [13, 18], and usually varies in time [1, 2, 14, 31, 50]. Investigation of the 1918 influenza pandemic in London, England, has shown that a time-varying transmission rate best explains the observed weekly mortality pattern [24]. For childhood diseases, temporal variation in the transmission rate is typically associated with school terms [17, 21, 22, 41, 50, 52]. In general, understanding how the transmission rate varies in time is vital to understanding the key factors that influence disease transmission, and is important for predicting future epidemics and evaluating control strategies [27].

A rich history of infectious diseases in humans has been preserved in the London Bills of Mortality and the Weekly Returns of the Registrar General's office, a data set that contains over 350 years of weekly mortality data in London, UK. This data set has recently been digitized and used to investigate the presence of herald waves preceding cholera epidemics [51], and to identify the seasonal pattern in smallpox transmission [37]. This invaluable resource presents the opportunity to study the dynamics of many different infectious diseases. Knowledge of the time-varying transmission rate is essential for determining which underlying factors caused changes in disease dynamics over the course of centuries. The potential to learn from this extensive data set is very exciting, but there are some practical limitations in reconstructing the transmission rate because of the length of the data set.

The recent literature describes several complex, sophisticated methods for estimating time-dependent disease transmission rates, including generalized profiling to fit deterministic models [28], and an explicit likelihood-based approach to fit stochastic models to partially observed Markov processes (POMP) [24, 26, 29, 35, 36]. These estimation methods are cast in a rigorous statistical framework, but are too computationally demanding to apply to time series longer than a few decades.

The purpose of this paper is to present a simple, fast, and intuitive estimation method for the time-varying transmission rate of an infectious disease, that can be applied to either incidence or mortality data. We build on the classic work of Fine and Clarkson [21] which uses a discrete-time approximation to infer the underlying transmission rate. Our estimation method is grounded in the standard SIR model [2,32–34,50] for infectious diseases. Pollicott *et al.* [46] have recently developed another fast method to estimate the time-varying transmission rate (from prevalence data) using the inverse problem for the SIR model. Future work should compare their estimation method with ours.

2 Three methods for estimating the time-varying transmission rate of an infectious disease

We will examine three successive 'fast' methods of estimating the *per capita* transmission rate $\beta(t)$. The notation used in the $\beta(t)$-estimation methods is recorded in Table 1 for easy reference. All three methods are derived from the standard SIR model (or SEIR model) with vital dynamics [2], *i.e.,*

$$\frac{dS}{dt} = \nu(t)N_0 - \beta(t)SI - \mu(t)S\,, \tag{1a}$$

$$\frac{dI}{dt} = \beta(t)SI - \gamma I - \mu(t)I\,, \tag{1b}$$

$$\frac{dR}{dt} = \gamma I - \mu(t)R\,, \tag{1c}$$

where S, I, and R are the numbers of individuals that are susceptible, infectious, or removed (either recovered or dead from the disease). $\nu(t)$ is the *per capita* birth rate, $\mu(t)$ is the *per capita* natural mortality rate, N_0 is the population at the initial time, t_0, and γ is the rate of removal from the infectious class due to recovery or death from disease (hence γ^{-1} is the mean infectious period). The SIR model (1) is based on the law of mass action, which assumes that the population is well mixed and that a certain proportion of contacts between infected and susceptible individuals results in new cases of infection [2, p. 65].

There are often concerns with using an SIR model, since the true infectious period of a disease is not distributed exponentially [40]. Additionally, the SEIR model is commonly used since it incorporates a latent class to account for individuals who have been infected but are not yet infectious [2, p. 663]. Krylova & Earn [38] have shown that for a fixed mean latent and infectious period, the dynamics of the SEIR model are not sensitive to the distribution of the infectious and latent periods and that for a fixed mean generation time, the dynamics of the SIR and SEIR model are almost identical. (The mean generation time of a disease, sometimes called the serial interval, is defined to be the time from initial infection of a primary case to initial infection of a secondary case [20].) The mean generation time in the SEIR model is the sum of the mean latent and infectious periods [38], so we set γ^{-1} in the SIR model to be the sum of the true mean latent and infectious periods to ensure the SIR model exhibits dynamics similar to the SEIR model.

2.1 The S method

First, we review the $\beta(t)$ estimation method presented by Fine & Clarkson [21] (referred to here as the "S method"). Let Z_t be the number of new infectious cases in the population from time $t - \Delta t$ to t. (Fine & Clarkson estimated Z_t for measles in England and Wales $(1950 - 1965)$ from case notification data by dividing the data by a two-thirds reporting ratio [21, p. 10].) Let S_t be the number of susceptible individuals at time t. If we assume that the generation time (*i.e.,* serial interval [20]) of all infected individuals is exactly Δt,

Symbol	**Definition**
Δt	Time interval between published case notifications (observation interval)
Z_t	Number of infections from time $t - \Delta t$ to time t
S_t	Number of susceptible individuals at time t
I_t	Number of infected individuals at time t
B_t	Number of births from time $t - \Delta t$ to time t
C_t	Number of cases reported from time $t - \Delta t$ to time t
T_{rep}	Mean time from initial infection to reporting (must be equal to $k\Delta t, k \in \mathbb{N}$)
$[T_{\text{inf}}]$	Mean time from initial infection to recovery (must be equal to $k\Delta t, k \in \mathbb{N}$)
η	Case fatality ratio (relevant if using mortality data)
ρ	Reporting ratio (proportion of cases that are reported)

Table 1: Notation used in the derivation of estimates for the transmission rate $\beta(t)$.

then currently infectious individuals will have infected their secondary cases and recovered a time Δt in the future. Thus we can relate the current and future number of infections using the mass action principle,

$$Z_{t+\Delta t} = Z_t S_t \beta_t \Delta t. \tag{2}$$

S_t increases as individuals are born, and decreases as susceptibles become infected. Let B_t be the number of births from time $t - \Delta t$ to t. We can keep track of the number of susceptibles by accounting for these births and infections in each time step,

$$S_{t+\Delta t} = S_t - Z_{t+\Delta t} + B_t. \tag{3}$$

Rearranging Equation (2), and keeping track of the susceptible population using Equation (3), Fine & Clarkson [21] estimated $\beta(t)$ as

$$\beta_t = \frac{Z_{t+\Delta t}}{Z_t S_t \Delta t}. \tag{4}$$

(In Fine & Clarkson's derivation they assume the time unit is chosen such that $\Delta t = 1$.) The dynamics of the discrete-time epidemic model formed by Equations (3) & (4) has been further analyzed in [43].

One disadvantage of this method is that the observation interval, Δt, is constrained to be equal to the generation time, which generally requires aggregation of incidence data to ensure the time between data points is equal to the generation time (*e.g.,* Fine & Clarkson [21] aggregated their data bi-weekly since the mean generation time for measles is approximately 2 weeks). Also, there is no natural mortality term in the susceptible update equation (3), so over a long period of time it is possible for the number of susceptibles to grow without bound.

With a little more effort, the observation interval constraint can be lifted and the estimation process tied much more closely to the SIR model.

2.2 The S^+ method

In her PhD thesis, Krylova modified the S method in order to estimate the transmission rate for smallpox in London, England over 250 years using mortality data from the London Bills of Mortality [37]. Instead of simply using the mass action principle to estimate $\beta(t)$, she derived an estimate of $\beta(t)$ explicitly from the SEIR model (referred to here as the "S^+ method"). The following is a summary of her derivation of the S^+ method [37], adapted slightly to apply to the simpler SIR model (1).

As in the S method, let S_t be the number of susceptibles in the population, B_t the number of births from time $t - \Delta t$ to t, and Z_t the number of new infections from $t - \Delta t$ to t. Additionally, let μ_t be the per capita natural mortality rate at time t. In the S^+ method, the observation interval, Δt is no longer constrained to be the generation time. The number of susceptibles in the population can be estimated using a discrete-time approximation to

the susceptible rate of change equation in the SIR model (1),

$$S_{t+\Delta t} = S_t + B_t - Z_{t+\Delta t} - \mu_t \Delta t S_t. \tag{5}$$

This equation is similar to equation (3) of the S method, but includes a natural mortality term.

Since Z_t is the number of new infections that have occurred in the last time interval Δt, we can keep track of Z_t using the SIR model by counting the cumulative number of individuals that enter the infectious class from time $t - \Delta t$ to t:

$$Z_t = \int_{t-\Delta t}^{t} \beta(\tau) S(\tau) I(\tau)\ d\tau. \tag{6}$$

Alternatively, if $[T_{\text{inf}}]$ is the mean time from initial infection to recovery, we can approximate Z_t by counting the number of individuals that leave the infectious class at time $[T_{\text{inf}}]$ in the future:

$$Z_t = \int_{t-\Delta t}^{t} (\gamma + \mu(\tau + [T_{\text{inf}}]))I(\tau + [T_{\text{inf}}])\ d\tau. \tag{7}$$

Here $[T_{\text{inf}}]$ is taken to be the mean generation time (γ^{-1}), rounded to the nearest Δt. If Δt is short enough that we can assume $\beta(t)$, $S(t)$, $I(t)$ and $\mu(t)$ are approximately constant over Δt, equations (6) and (7) can be rewritten as

$$Z_t \approx \beta_t S_t I_t \Delta t, \tag{8}$$

and

$$Z_t \approx (\gamma + \mu_{t+[T_{\text{inf}}]}) I_{t+[T_{\text{inf}}]} \Delta t. \tag{9}$$

Rearranging equation (9), I_t can be approximated as follows:

$$I_t \approx \frac{Z_{t-[T_{\text{inf}}]}}{(\gamma + \mu_t)\Delta t}. \tag{10}$$

Then replacing I_t in equation (8) with the right side of (10),

$$Z_t \approx \beta_t S_t \frac{Z_{t-[T_{\text{inf}}]}}{\gamma + \mu_t}. \tag{11}$$

Rearranging equation (11) provides an estimate of β_t,

$$\beta_t = \frac{1}{S_t} \frac{Z_t}{Z_{t-[T_{\text{inf}}]}} (\gamma + \mu_t) \tag{12}$$

which differs from Equation (4) of the S method by the factor $(\gamma + \mu_t)$.

Usually we do not observe the exact number of new cases in a time period; instead we have case notification data that provides a sample of the true case count. The S^+ method can be applied to case notification data, C_t, via

$$C_t = \rho\eta Z_{t-T_{\text{rep}}}, \qquad T_{\text{rep}} \in \{\Delta t, 2\Delta t, ...\}, \tag{13}$$

where ρ is the proportion of cases (or deaths) that are reported, η is the case fatality ratio if we are dealing with mortality data (η is set to 1 otherwise), and T_{rep} is the delay between infection and reporting. The time T_{rep} is an integer multiple of the observation interval Δt, because T_{rep} tells us how many points forward in the case notification data we must look for a reported infection (we are assuming the delay T_{rep} is the same for every case).

In summary, the S^+ method uses equations (5) and (12) to estimate $\beta(t)$, derived explicitly from the SIR model (in contrast to equations (2) and (3) in the S method). The S^+ method avoids unnecessary aggregation of data, since the observation interval, Δt, can differ from the generation time. In addition it accounts for a delay in reporting and can be applied to mortality data.

Krylova [37] estimated the seasonality of transmission of measles in England and Wales (1950 - 1965) using weekly measles case notification data. She used the S^+ method to estimate the transmission rate $\beta(t)$ for the entire time series, and then divided this estimate

β_t by its long term trend to identify the seasonal component of β_t. Her estimate of the seasonality of $\beta(t)$ is qualitatively similar to the estimate of Fine and Clarkson [21] (the S method), and the estimate of Finkenstädt and Grenfell [22], who used a more sophisticated method of fitting a discrete-time stochastic SEIR model (the TSIR model) to the measles data set. She was also able to produce a qualitatively similar seasonal transmission estimate with the S^+ method as Hooker *et al.* [28] who used generalized profiling to fit a deterministic SEIR model to measles incidence data in Ontario, Canada (1939 - 1965).

2.3 The SI method

The SI method improves upon the S^+ method by additionally estimating the number of infectious individuals at each point in time using a discrete-time approximation to the infected rate of change equation in the SIR model (1) (in addition to the discrete-time approximation to the susceptible range of change equation). We will see that having an estimate of both the susceptible and infected populations at each point in time still allows for easy estimation of $\beta(t)$.

We define the following discrete-time approximations to the continuous SIR model (1), as an estimate of the number of susceptible and infected individuals at each point in time:

$$S_{t+\Delta t} = S_t + B_t - Z_{t+\Delta t} - \mu_t \Delta t S_t \tag{14}$$

$$I_{t+\Delta t} = I_t + Z_{t+\Delta t} - (\gamma + \mu_t)\Delta t I_t. \tag{15}$$

Equation (15) estimates $I(t)$ more accurately than the approximation made by the S^+ method in Equation (10). In equation (6) of the S^+ method, we know the number of cases from time $t - \Delta t$ to t (Z_t) via

$$Z_t = \int_{t-\Delta t}^{t} \beta(\tau) S(\tau) I(\tau)\ d\tau. \tag{16}$$

Once again, assuming that Δt is small enough that $\beta(t)$, $S(t)$, and $I(t)$, are approximately constant over Δt, we can approximate Z_t using the value for $\beta(t)$, $S(t)$, and $I(t)$ either at the left or right end point of the integral in equation (16).

If we use the right endpoint we have:

$$Z_t \approx \beta_t S_t I_t \Delta t \tag{17}$$

as in the S^+ method. If we use the left endpoint we have:

$$Z_t \approx \beta_{t-\Delta t} S_{t-\Delta t} I_{t-\Delta t} \Delta t. \tag{18}$$

Alternatively, we could take some linear combination of the endpoints to find an optimal approximation of Z_t. For example, we might take the average of $\beta(\tau)S(\tau)I(\tau)$ at each endpoint and hope to obtain a better approximation,

$$Z_t \approx \frac{1}{2}\left[\beta_t S_t I_t \Delta t + \beta_{t-\Delta t} S_{t-\Delta t} I_{t-\Delta t} \Delta t\right]. \tag{19}$$

Thus, $\beta(t)$ can be estimated by rearranging any of equations (17), (18), (19) to get

$$\beta_t = \frac{Z_t}{S_t I_t \Delta t} \qquad \text{(right endpoint)} \tag{20a}$$

$$\beta_t = \frac{Z_{t+\Delta t}}{S_t I_t \Delta t} \qquad \text{(left endpoint)} \tag{20b}$$

$$\beta_t = \frac{1}{2}\left[\frac{Z_t}{S_t I_t \Delta t} + \frac{Z_{t+\Delta t}}{S_t I_t \Delta t}\right]. \qquad \text{(average of endpoints)} \tag{20c}$$

where I_t and S_t are computed as in equations (14), (15). Once again, this can be applied to case notification data (C_t) using Equation (13).

Taking the left endpoint of the integral in Equation (16) (*i.e.,* computing β_t from Equation (20b)) performs the best by far when tested, since using the right endpoint results in a lagging β_t estimate (see §S4.4 of the Supplementary Material). So from this point forward,

we define the SI method to be the estimate of β_t obtained by using the left endpoint of the integral as in Equation (20b).

The recurrence relations for S_t and I_t in equations (14) and (15) can be solved (see §S4.5 of the Supplementary Material for a proof using mathematical induction). Having exact solutions to the recurrence relations slightly increases computational efficiency and facilitates some asymptotic analysis. To illustrate this point, consider the special case where the time period of interest is short enough that the natural mortality rate $\mu(t)$ is approximately constant. For many data sets the assumption that $\mu(t)$ is constant is reasonable as $\mu(t)$ changes over a much longer timescale than the epidemics occur (if we ignore seasonality in $\mu(t)$) [25]. Setting $t_0 = 0$, the solutions to the recurrence relations are then simply:

$$S_{j\Delta t} = S_0(1-\mu\Delta t)^j + \sum_{k=1}^{j}(1-\mu\Delta t)^{j-k}\left[B_{(k-1)\Delta t} - Z_{k\Delta t}\right] \qquad j = 1, 2, 3, .. \tag{21a}$$

$$I_{j\Delta t} = I_0(1-(\gamma+\mu)\Delta t)^j + \sum_{k=1}^{j}(1-(\gamma+\mu)\Delta t)^{j-k} Z_{k\Delta t} \qquad j = 1, 2, 3, ... \tag{21b}$$

which can be inserted in Equation (20b) to obtain an explicit estimate of $\beta(t)$ given an observed time series Z_t. (§S4.5 of the Supplementary Material provides the solution to the recurrence relations in the case of a non-constant $\mu(t)$.)

Equations (21a) and (21b) display the dependence of the estimates S_t and I_t on the initial conditions, S_0 and I_0. In the estimate of $I_{j\Delta t}$, the initial condition I_0 is multiplied by $(1-(\gamma+\mu)\Delta t)^j$. Given measles parameters in time units of a week (Table 2), and assuming $\Delta t = 1$ week this quantity is approximately 0.461^j. Thus, as time increases, the estimate $I_{j\Delta t}$ will rapidly have negligible dependence on the initial number of infected individuals. For example, after one year the contribution of I_0 to $I_{j\Delta t}$ is reduced by a factor $0.461^{52} \approx 3.25 \cdot 10^{-18}$.

By contrast, in the estimate of $S_{j\Delta t}$, the initial condition S_0 is multiplied by $(1-\mu\Delta t)^j$. For the above parameters, this quantity is approximately 0.999^j. Thus, after one year $S_{j\Delta t}$ depends on a reduced factor of $0.999^{52} \approx 0.949$ of S_0. Thus the estimate of S_t depends

strongly on the initial condition, S_0, but the estimation of I_t depends little on the initial condition I_0. Generally, the initial conditions are parameter values that are extremely hard to estimate for a data set, and an incorrect estimate of S_0 strongly affects the estimate of $\beta(t)$. Compared with the S^+ method, the only new parameter value in the SI method is I_0, and fortunately an incorrect estimate of I_0 has a negligible effect on the estimation of $\beta(t)$.

The S, S^+, and SI methods are all extremely fast computationally. For example, estimating the transmission rate with the SI method from weekly case notification data for 300 years takes less than 0.5 seconds on a MacBook Pro laptop with a 2 GHz Intel Core i7 processor.

3 Comparing the performance of the three estimation methods

3.1 Simulating case notification data

In order to compare the performance of the three methods, we test each estimation method on simulated case notification data. To generate these data, we define the SIRQ model to be the set of SIR equations in Equation (1) with an additional differential equation for the cumulative number of cases, $Q(t)$, from the initial time t_0 to time t,

$$\frac{dS}{dt} = \nu(t)N_0 - \beta(t)SI - \mu(t)S\,, \tag{22a}$$

$$\frac{dI}{dt} = \beta(t)SI - \gamma(t)I - \mu(t)I\,, \tag{22b}$$

$$\frac{dR}{dt} = \gamma(t)I - \mu(t)R\,, \tag{22c}$$

$$\frac{dQ}{dt} = \beta(t)SI\,. \tag{22d}$$

Then using the `deSolve` package [49] in `R` [47], we numerically solve the SIRQ model (Equation (22)) given a set of parameters, initial conditions, and the chosen transmission rate $\beta(t)$.

As in Equation (13), let C_t represent typical case notification data with a time interval Δt between published notifications, *i.e.,* C_t is the total number of reported cases since the last observed time point. If a fixed proportion $\rho\eta$ of cases are reported, C_t can be computed from the numerical SIRQ solution (22) via

$$C_{t-T_{\text{rep}}} = \rho\eta\Big(Q(t) - Q(t-\Delta t)\Big). \tag{23}$$

3.1.1 A sinusoidally forced transmission rate

In our simulations we use a sinusoidally forced transmission rate $\beta(t)$ (as in [3,13,30,41,44]), with mean β_0 and amplitude α, *i.e.,*

$$\beta(t) = \max\left\{\beta_0\left[1+\alpha\left(\cos\left(\frac{2\pi t}{Y}\right)+\epsilon\phi(t)\right)\right], 0\right\}, \tag{24}$$

where the constant Y is one year in the chosen time unit, $\phi(t)$ is a point drawn from a Normal$(0,1)$ distribution, and ϵ is the intensity of the 'noise' term $\phi(t)$. In Equation (24) we take the maximum of the sinusoid and zero to ensure $\beta(t)$ is never negative. However, in practice for $\epsilon \leq 0.5$ and $\alpha \leq 0.1$ such negative values do not occur anyway.

Term-time forcing is also commonly used for the transmission rate [8, 18, 30, 48], but Earn *et al.* [18, References and Notes: #13] found that the dynamics of the SEIR model is qualitatively equivalent for a term-time forced transmission rate with amplitude α_1 and a sinusoidally forced transmission rate with amplitude α_2, for some $\alpha_2 < \alpha_1$.

In the SIR model with constant vital dynamics ($\mu(t) = \nu(t) \equiv \mu$) and a constant $\beta(t) \equiv \beta_0$, the basic reproduction number, $\mathcal{R}_0$, is [38, Supplementary Material (S6)], [42]

$$\mathcal{R}_0 = \frac{\beta_0 N_0}{\gamma+\mu}, \tag{25}$$

so the mean value of $\beta(t)$ can be written

$$\beta_0 = \frac{\mathcal{R}_0(\gamma+\mu)}{N_0}. \tag{26}$$

3.1.2 Initial conditions

Since the solutions of the sinusoidally forced SIR model oscillate around the equilibrium value of the unforced model with $\beta(t) \equiv \beta_0$, we chose the initial conditions for $S(t)$ and $I(t)$ to be the endemic equilibrium values of the unforced SIR model with constant vital dynamics, $\mu(t) = \nu(t) \equiv \mu$ (derivation in §S2.1 of the Supplementary Material),

$$S_0 = \hat{S} \equiv \frac{N_0}{\mathcal{R}_0}, \tag{27a}$$

$$I_0 = \hat{I} \equiv N_0 \left(1 - \frac{1}{\mathcal{R}_0}\right) \frac{\mu}{\gamma+\mu}. \tag{27b}$$

3.2 Computing the error in estimation

We have simulated many case notification time series and used the S, S^+, and SI methods to estimate $\beta(t)$. As a measure of accuracy we calculate the relative root mean square error (RRMSE) between the true continuous $\beta(t)$ and the estimated discrete $\beta_{j\Delta t},\ j = 1, 2, ...n$ using:

$$\text{RRMSE} = \sqrt{\frac{\sum_{j=1}^{n} \left(\beta(j\Delta t) - \beta_{j\Delta t}\right)^2}{n\left[\overline{\beta(t)}\right]^2}}, \tag{28}$$

where n is the number of time points in β_t, and $\overline{\beta(t)}$ is the mean value of the true $\beta(t)$ at the observation times, *i.e.,*

$$\overline{\beta(t)} = \frac{1}{n}\sum_{j=1}^{n} \beta(j\Delta t)\,. \tag{29}$$

Symbol	Definition	Measles	Smallpox	Source
$\mu(t)$	Yearly *per capita* natural mortality rate	**0.04** $[0.01 - 0.16]$	**0.04** $[0.01 - 0.16]$	In London, England, $\mu = 0.045$ yr^{-1} in 1662 (estimated from the London Bills of Mortality [37, p. 154]), and $\mu = 0.026 - 0.048$ yr^{-1} from 1700 - 1820 [39].
$\nu(t)$	Yearly *per capita* birth rate	**0.04** $[0.01 - 0.16]$	**0.04** $[0.01 - 0.16]$	$\nu(t)$ is set equal to $\mu(t)$
N_0	Population at time t_0	$\mathbf{500,000}$	$\mathbf{392,400}$	Population of London, England in 1662 for smallpox (when the London Bills of Mortality begin) and in 1700 for measles [37, p. 108].
γ^{-1}	Mean disease generation time (in days)	**13** $[3 - 52]$	**22** $[6 - 88]$	γ^{-1} is taken to be the sum of the mean infectious and latent periods of the disease (see discussion of the mean generation time in §2). The mean latent period is 15 days for smallpox; 8 days for measles, and the mean infectious period is 7 days for smallpox; 5 days for measles [2, 19, 37] .
S_0	Initial condition for $S(t)$	$\mathbf{25,000}$ $[6,250 - 100,000]$	$\mathbf{98,100}$ $[24,525 - 392,400]$	Computed using Equation (27a)
I_0	Initial condition for $I(t)$	**678** $[170 - 2,712]$	**710** $[178 - 2,840]$	Computed using Equation (27b)
ρ	Reporting ratio	**1** $[0.01, 1]$	**1** $[0.01, 1]$	Except for the the section dealing with observation error (§4.3), the reporting ratio is set to be 1, since this factor just scales the data and estimate of $\beta(t)$. For smallpox in the London Bills of Mortality the reporting ratio was close to 100% as smallpox was an easily recognizable disease [19].
η	Case fatality ratio	**1** $[0.01, 1]$	**1** $[0.01, 1]$	As for the reporting ratio, the case fatality ratio is set to 1. For smallpox the case fatality ratio is estimated to be 20% during the London Bills of Mortality [19,37]. For measles, the case fatality ratio was 2.5% in the US in 1912, 0.3% in the US from 1987–2000, and between 3% and 34% in developing countries from 1970–1980 [45].
$\mathcal{R}_0$	Basic reproduction number	**20** $[2 - 30]$	**4** $[2 - 30]$	$\mathcal{R}_0$ for smallpox is between 3 and 6, [2]; $\mathcal{R}_0$ chosen here to be 4 as in [37]. For measles, $\mathcal{R}_0 \approx 20$ [16], $\mathcal{R}_0 = 16 - 18$ in England and Wales (1950 - 1958), [2, p. 70] $\mathcal{R}_0 = 13 - 14$ in Cirencester, England (1947 - 50), [2, p. 70].
α	Amplitude of sinusoidal $\beta(t)$	**0.08** $[0 - 0.1]$	**0.08** $[0 - 0.1]$	$\alpha = 0.032 - 0.12$ in London, England from 1665 - 1930 for smallpox [37], and $\alpha = 0.08$ for measles in [18]. Note: $\alpha = 0.08$ with a sinusoidally forced $\beta(t)$, corresponds to $\alpha = 0.25$ with a term time forced $\beta(t)$ [18].
Δt	Time interval between published case notifications	**1** week	**1** week	The London Bills of Mortality contain weekly data.

Table 2: Parameters for measles and smallpox in London, England from $1662 - 1930$. Both the estimated parameter value (in **bold**) and the range explored in this paper [in grey] are listed.

4 Results

From a given simulation generated by Equation (22) with measles parameters (Table 2), we estimate the transmission rate $\beta(t)$ with each of the S, S^+ and SI methods (see Figure 1). It is clear that the S method is much less accurate in estimating $\beta(t)$ than the S^+ or SI methods, in part because it was not derived explicitly from the SIR model (from which the simulated case notification data is generated). Since the susceptible update equation (3) in the S method does not contain any natural mortality, the number of susceptibles steadily increases over time, causing the estimated β_t to steadily decrease. Also, the S method requires rounding of the mean generation time (13 days for measles) to the nearest Δt (1 week) which causes underestimation of $\beta(t)$ even at the beginning of the time series. The S^+ method is a revised version of the S method that corrects for these errors, so we will proceed with only considering the S^+ and SI methods. In the case of measles parameters, the SI method outperforms the S^+ method, having almost half the error in estimation compared to the S^+ method. This is especially relevant in the case of a noisy $\beta(t)$ as the error is generally much larger.

4.1 Dependence on parameter values

While it is valuable to check how each method performs for the estimated measles parameters stated in Table 2, it is important to ensure that the estimation methods perform well when applied in a more general context. We now consider how well the S^+ and SI methods estimate $\beta(t)$ for a range of parameter values.

In order to explore the parameter space, we start with a 'base' set of either measles or smallpox parameters. Then, one at a time, we vary a parameter up to a factor of 4, from 25% to 400% of its value in the 'base' parameter set. Measles and smallpox parameter sets, and the 25% to 400% parameter ranges for $\mu(t), \nu(t), N_0, \gamma^{-1}, S_0$, and I_0 are listed in Table 2. For each parameter value in the range, the simulated case notification data is generated

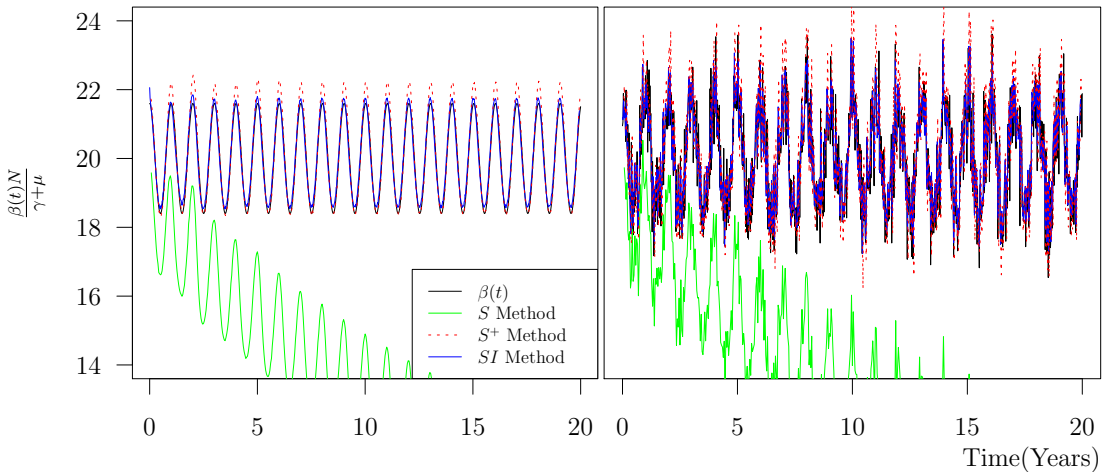


Figure 1: An example of estimating the transmission rate $\beta(t)$ using the S, S^+ and SI methods (§2) for 20 years with measles parameters (Table 2). Each panel contains the $\beta(t)$ used to simulate the case notification data, as well as the estimates from each method. $\beta(t)$ is plotted in units of $\mathcal{R}_0$. In the left panel, the true $\beta(t)$ is constructed using equation (24) with $\mathcal{R}_0 = 20, \alpha = 0.08, \epsilon = 0$, and the relative root mean square error (RRMSE, Equation (28)) for each of the estimation methods is $(S, S^+, SI) = (0.34, 0.014, 0.008)$. In the right panel, the true $\beta(t)$ is constructed with $\mathcal{R}_0 = 20, \alpha = 0.08, \epsilon = 0.5$, and the RRMSE for each of the estimation methods are $(S, S^+, SI) = (0.34, 0.062, 0.032)$.

(Equation (22)) with a sinusoidal $\beta(t)$ ($\alpha = 0.08$, and $\epsilon = 0$ in Equation (24)), and $\beta(t)$ is estimated using the S^+ and SI method.

Figure 2 displays how each method performs for a range of γ, μ, ν, S_0, and I_0. For the explored range of parameters, $\beta(t)$ is usually estimated accurately with error less than 4% of its mean value. The SI method (§2.3) estimates $\beta(t)$ with greater accuracy than the S^+ method (§2.2) for all explored parameters. In part since $\mathcal{R}_0$ for smallpox is small, $\beta(t)$ is estimated more accurately for smallpox parameters than measles parameters (dependence of estimation accuracy on $\mathcal{R}_0$ is explored later in Figure 3).

As noted in §3.1.2, since the solutions of the sinusoidally forced SIR model oscillate around the equilibrium $(\hat{S}, \hat{I})$ value of the unforced model (constant $\beta(t) \equiv \beta_0$) it is reasonable to set the initial conditions (S_0, I_0) in our simulations to be $(\hat{S}, \hat{I})$. Accuracy in estimation of $\beta(t)$ depends strongly on the initial number of susceptibles S_0. If S_0 is far from $\hat{S}$, $\beta(t)$ is estimated poorly until $S(t)$ settles back to oscillating about $\hat{S}$. If S_0 is much larger than $\hat{S}$, $I(t)$ initially increases very rapidly, depleting the susceptible population, and then crashes

dramatically because there are virtually no susceptibles left. These sudden changes in $S(t)$ and $I(t)$ make estimation methods for these quantities less effective. The estimation error for very large or small S_0 values is beyond the range plotted in Figure 2 and for the SI method with measles parameters, reaches a maximum RRMSE of 0.137 when $S_0 = 4\hat{S}$. Estimation of $\beta(t)$ is not sensitive to I_0, as expected from the solution to the recurrence relations (Equation (21b)).

The mean generation time (γ^{-1}) is also a parameter that substantially affects the error in estimation of $\beta(t)$. Error in the SI method decreases exponentially as γ^{-1} increases, whereas error in the S^+ method oscillates as a function of γ^{-1}, increasing both for large and small values of γ^{-1}. The oscillation can be attributed to rounding the parameter $[T_{\text{inf}}]$, used in the S^+ method (Equation (12)). $[T_{\text{inf}}]$ is the mean time from infection to recovery, rounded to the nearest Δt, which we have taken to be a week.

In the measles base parameter set (Table 2), estimation accuracy appears to be sensitive to the birth rate $\nu(t)$, with a maximum error in estimation occurring when $\nu(t)$ is twice as large as its value in the measles parameter set. In that case, both the S^+ and SI estimates of $\beta(t)$ are good approximations, except at the peak of the true $\beta(t)$ where there is a small dip in each of the estimates. This dip is due to an overestimation of $I(t)$ at the peak of transmission (demonstrated in §S6.1.2 of the Supplementary Material). In the smallpox parameter set we instead see that estimation error increases as the disparity between $\nu(t)$ and $\mu(t)$ increases, and does not depend on just the value of $\nu(t)$ itself. Because estimation error depends differently on $\nu(t)$ for the two parameter sets, the parameter space would have to be explored further to understand the dependence of estimation accuracy on $\nu(t)$.

Figure 2 demonstrates the dependence of each method's accuracy on parameter values for a smooth $\beta(t)$ ($\epsilon = 0$ in Equation (24)). §S6.1.1 of the Supplementary Material contains the same figure but for a 'noisy' $\beta(t)$ ($\epsilon = 0.5$ in Equation (24)). In that case, the dependence on each parameter is similar, although the estimation error is higher overall due to the noise. In this 'noisy' $\beta(t)$ case, the S^+ method has twice as much estimation error as the SI method.

When testing the estimation methods, we chose a transmission rate $\beta(t)$ to simulate case notification data, and then estimated $\beta(t)$ with each of the methods. The mean and amplitude of the true $\beta(t)$ influences the success of our reconstruction. For a range of $\mathcal{R}_0$ and seasonal amplitude α, Figures 3–4 display the estimation accuracy of each method for $\epsilon = 0$ and $\epsilon = 0.5$ respectively (using measles parameters in Table 2). Estimation error increases as $\mathcal{R}_0$ and α increase, and more than doubles in the presence of noise in $\beta(t)$. For noisy $\beta(t)$ the error in estimation is much larger for large values of α since noise is added to $\beta(t)$ in proportion to α. The same figures are produced for smallpox parameters in §S6.2.1 of the Supplementary Material and are qualitatively identical.

The Supplementary Material also contains figures that demonstrate the error in estimation of $\beta(t)$ from simulations with much larger seasonal amplitude $\alpha \in [0, 0.9]$ (§S6.2.2). For large amplitude α, estimation of the transmission rate becomes less accurate and the SI method estimates $\beta(t)$ much more accurately than the S^+ method.

4.2 Sensitivity to incorrect parameter values

Some of the parameters necessary for estimating the transmission rate of an infectious disease are typically poorly known (*e.g.,* reporting ratio). It is therefore important to know how well each estimation method performs if the estimate is calculated with the wrong parameter values. In this section, $\beta(t)$ and the simulated case notification data are generated using 'correct' parameter values, and each method estimates $\beta(t)$ using one 'incorrect' parameter value. For both smallpox and measles parameters, we explore each method's sensitivity to incorrect parameters, by varying each parameter by a factor of two from its 'correct' value when estimating $\beta(t)$.

Estimating $\beta(t)$ with the wrong parameter values yields errors that are much larger than the difference in error between the S^+ and SI methods. Consequently, the sensitivity of both methods to incorrect parameter values looks qualitatively the same.

Figure 5 shows that having incorrect information about the birth rate, ν, and the re-

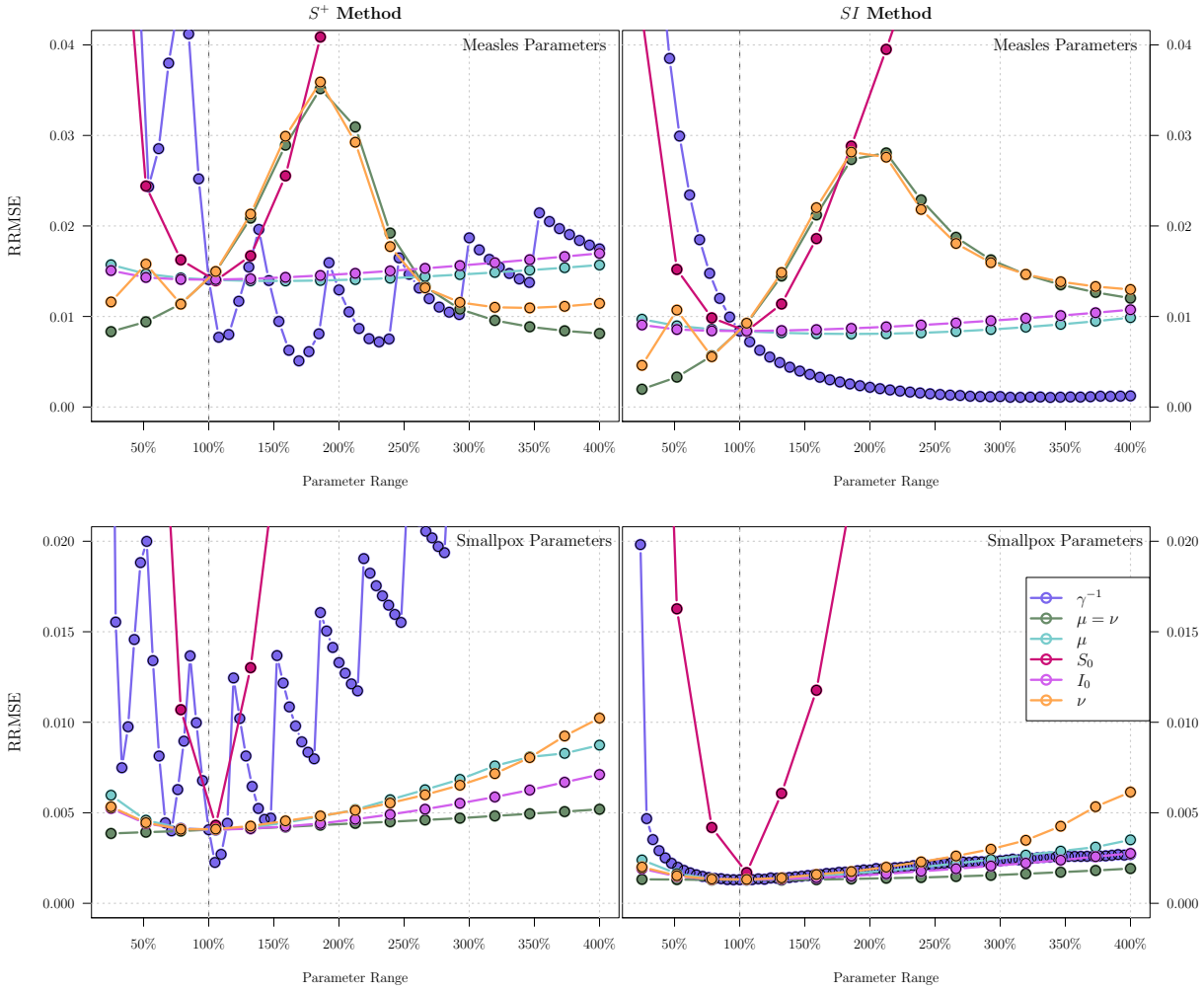


Figure 2: Accuracy in estimating the transmission rate $\beta(t)$ using the S^+ and SI methods (§2.2 & §2.3) for a variety of parameter values. Each point plotted represents the relative root mean square error (RRMSE, Equation (28)) in estimating $\beta(t)$ for one set of parameter values. For each parameter set, case notification data is simulated with a chosen $\beta(t)$ (Equation (24) with $\alpha = 0.08, \epsilon = 0$) and then used to estimate $\beta(t)$ with the S^+ or SI method.

The top two panels begin with measles parameters at the 100% parameter range line, and then one parameter at a time is varied from 25% to 400% of its value in the measles parameter set. The bottom two panels are the same but for smallpox parameters. Parameter definitions, estimates and ranges are stated in Table 2.

For the range of parameters explored here, $\beta(t)$ is usually estimated accurately within 4% of its mean value. The SI method estimates $\beta(t)$ more accurately than the S^+ method in all cases. Estimation accuracy decreases rapidly if the susceptible initial condition (S_0) is far from the equilibrium of the unforced SIR model (see discussion in §4.1). In the SI method, estimation error decreases for larger mean generation times (γ^{-1}), and in the S^+ method, both small and large γ^{-1} increase estimation error. Estimation accuracy does not depend strongly on the birth rate (ν), the natural mortality rate (μ), the disparity between the birth and natural mortality rates, or the infected initial condition (I_0).

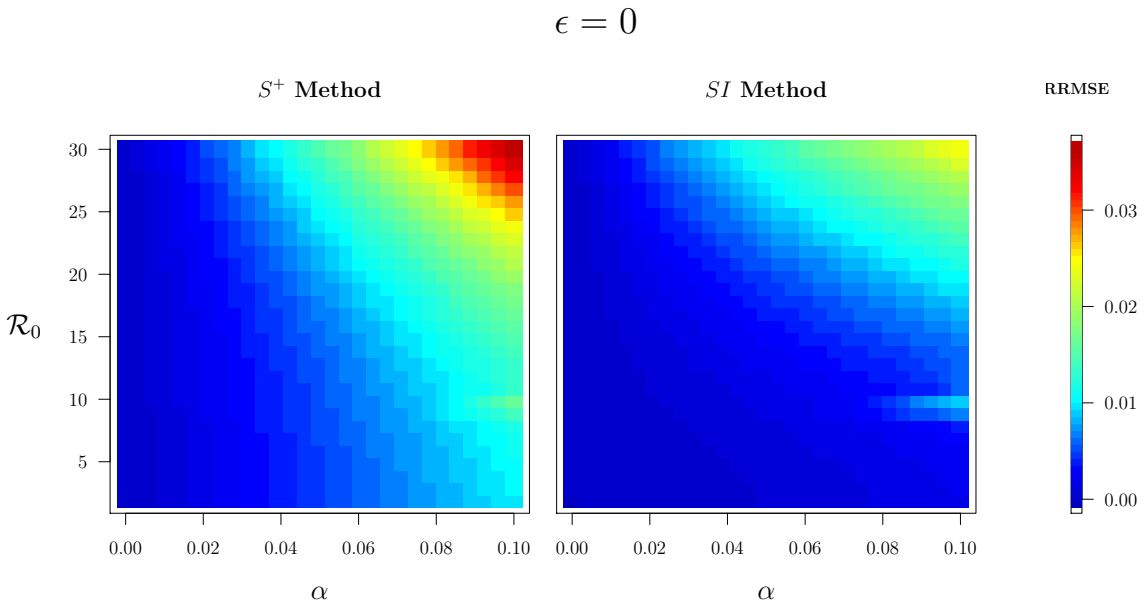


Figure 3: Dependence of the performance of the S^+ and SI estimation methods (§2.2 & §2.3) on the underlying basic reproduction number $\mathcal{R}_0$ and the seasonal amplitude in transmission α. For each α-$\mathcal{R}_0$ pairing, the transmission rate $\beta(t)$ is computed (using Equation (24) with $\epsilon = 0$), and used to simulate case notification data with measles parameters (Table 2). Then the S^+ and SI methods estimate $\beta(t)$ using the simulated data, and the relative root mean square error (RRMSE, Equation (28)) between the true $\beta(t)$ and the estimated β_t is calculated.

In general the SI method estimates $\beta(t)$ with greater accuracy than the S^+ method. Larger values of α and $\mathcal{R}_0$ decrease estimation accuracy, though the estimation error is still small ($< 4\%$ of the mean value of $\beta(t)$) for the range of α and $\mathcal{R}_0$ explored here ($\alpha \in [0, 0.1], \mathcal{R}_0 \in [2, 30]$).

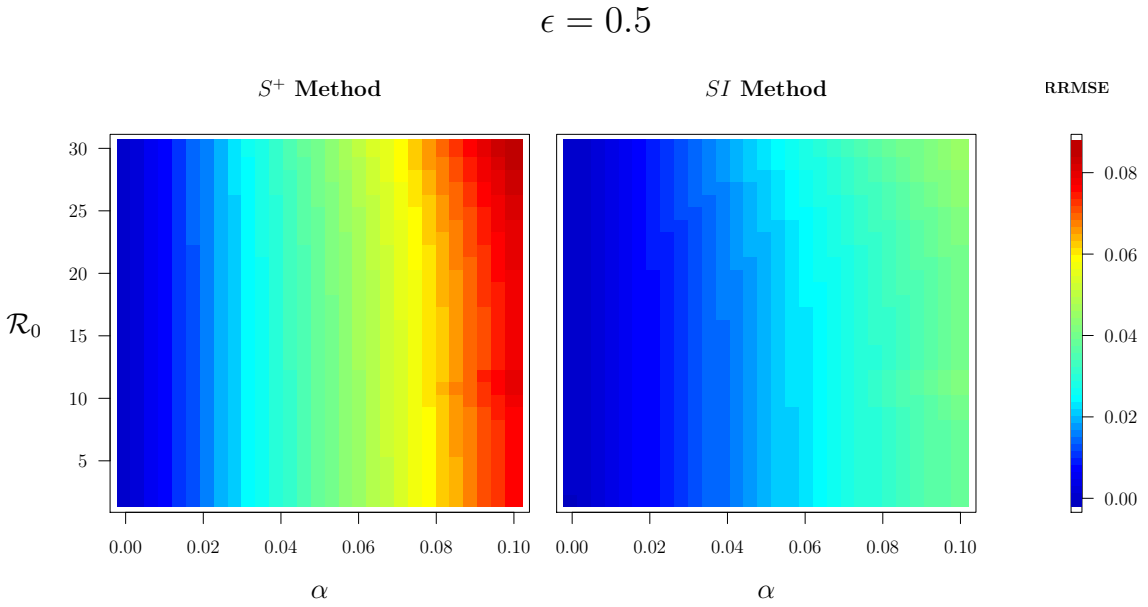


Figure 4: Dependence of estimation accuracy on the basic reproduction number $\mathcal{R}_0$ and the seasonal amplitude in transmission α as in Figure 3 but with a 'noisy' transmission rate $\beta(t)$ ($\epsilon = 0.5$ in Equation (24)). Once again, the SI method estimates $\beta(t)$ with greater accuracy than the S^+ method. Larger α and $\mathcal{R}_0$ decrease the estimation accuracy, but increasing α causes a more substantial accuracy reduction. This is because noise is added to $\beta(t)$ as a proportion of the amplitude α. So larger α values result in a noisier transmission rate to estimate. Even with the noisy transmission rate, the error in estimation is small ($< 9\%$ of the mean value of $\beta(t)$) for the range of $\mathcal{R}_0$ and α explored here.

porting/case fatality ratio, $\rho\eta$, causes substantial error in estimating $\beta(t)$. If our estimate for ν is twice as large as the true value for ν, our error is ten times as large, and even worse if ν is underestimated. Underestimating ν, or $\rho\eta$ is essentially equivalent to adding fewer people to the susceptible class than are being removed, so the number of susceptibles in the update equation (14) eventually becomes negative. This is helpful information practically as it allows us to constrain our parameter estimates; we know we have underestimated ν or $\rho\eta$ if the estimated number of susceptibles has become negative.

Estimation of $\beta(t)$ is also sensitive to incorrect parameter values for γ^{-1}, μ, and S_0, but is not sensitive to to incorrect parameter values for I_0 or T_{rep}. Once again, the lack of sensitivity to the initial number of infectives is not surprising because the estimate I_t depends very weakly on the initial condition I_0 (as seen in Equation (21b)).

Overall, Figure 5 shows that estimation of parameter values is very important for accurate estimation of the transmission rate. It is unfortunate that both methods are so sensitive to $\rho\eta$, since it is difficult to estimate the reporting ratio of a data set. Further work involving a method for fitting $\rho\eta$ to the data set is clearly necessary.

4.3 Sensitivity to observation error

The real world provides much noisier data than the case notification data C_t simulated by the SIR model. Up to this point we have tested the $\beta(t)$ reconstruction methods using C_t where $C_t = \rho\eta Z_{t-T_{\text{rep}}}$, which is the number of new infections, $Z_{t-T_{\text{rep}}}$, multiplied by the proportion of cases that are reported (ρ), and the proportion of cases that result in death (η). In this case, $\rho\eta$ represents the proportion of infections that are recorded in the data set, and for convenience we will consider them together as one quantity that represents the reporting/case fatality ratio. We could more accurately simulate what happens in the world by assuming the case notification data is sampled from a binomial distribution, where the number of trials is the number of new infections over time Δt, and the probability that an infection is recorded (the probability of 'success') is equal to the reporting/case fatality ratio

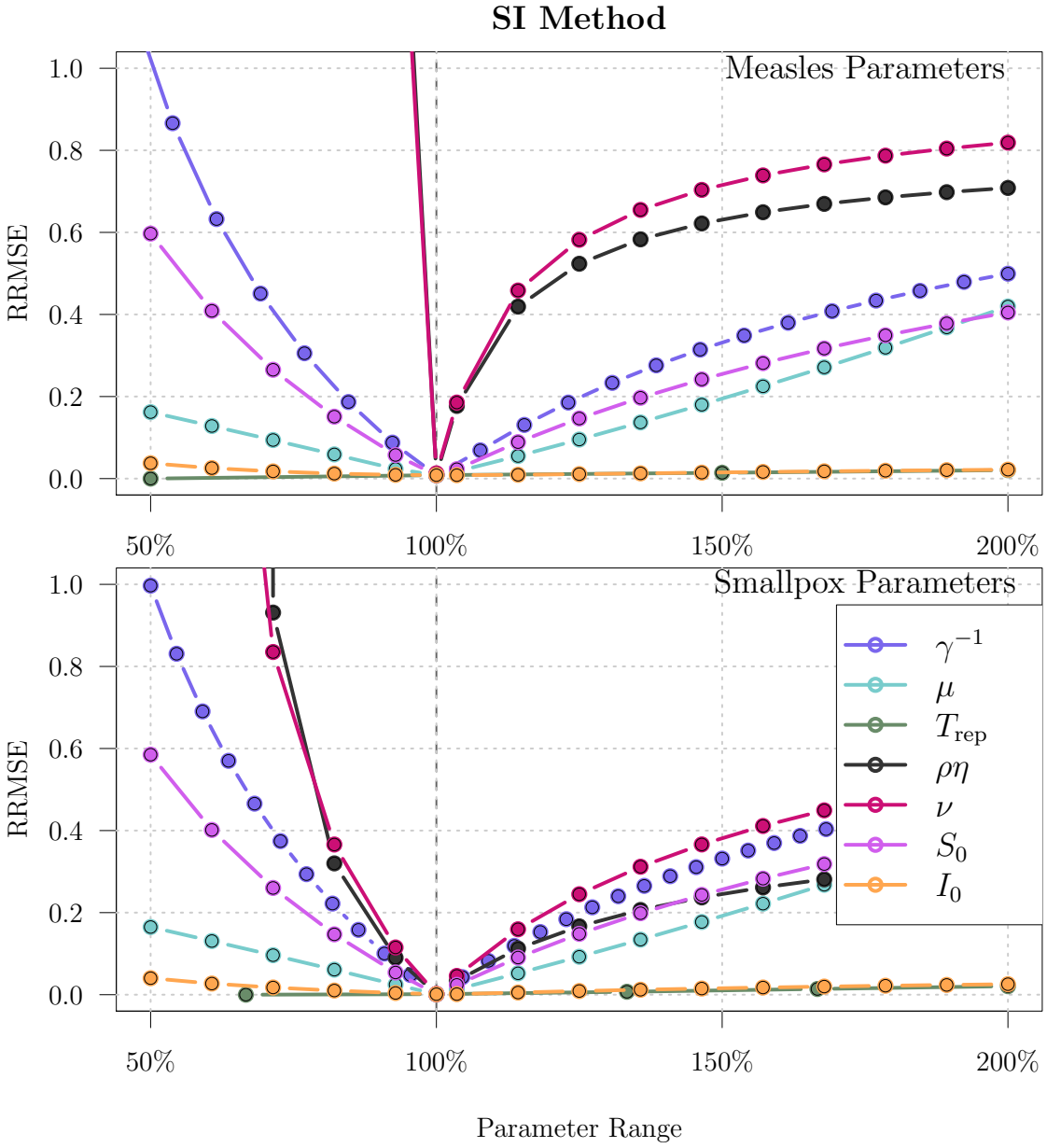


Figure 5: Sensitivity to incorrect parameter values when estimating the transmission rate $\beta(t)$. Each point represents the relative root mean square error (RRMSE, Equation (28)) in estimating $\beta(t)$ with the SI method (§2.3) for a set of parameter values, where one parameter is incorrect. Given case notification data simulated from the SIR model and measles (top panel) or smallpox (bottom panel) parameters (see Table 2), $\beta(t)$ is estimated from a set of parameters where one parameter varies from $50 - 200\%$ of the value that was used to generate the simulated data. The vertical line at 100% marks the set of true parameter values used to simulate the case notification data.

Estimating $\beta(t)$ with incorrect parameters yields error that is much larger than the difference between the S^+ and SI methods, so qualitatively this plot looks identical for the S^+ method (this plot is produced for the S^+ method in §S7 of the Supplementary Material). Accurate estimation of $\beta(t)$ is most sensitive to an underestimate of the birth rate (ν) and the reporting/case fatality ratio ($\rho\eta$), but is also sensitive to the mean generation time (γ^{-1}), the initial number of susceptibles (S_0), and the natural mortality rate (μ). Accurate estimation of $\beta(t)$ is not sensitive to the initial number of infected individuals (I_0) or the time from infection to reporting (T_{rep}).

($\rho\eta$).

Figure 6 shows the accuracy of each estimation method if the case notification data is sampled from the binomial distribution, with $\rho\eta = 0.2$ and measles parameters. As in Figures 3–4, larger $\mathcal{R}_0$ and α values cause greater error in estimation for both methods. However, in the presence of observation error a low $\mathcal{R}_0$ value of 2 or 3 also results in large error in estimation. With observation error, we see that the SI method estimates $\beta(t)$ much more accurately than the S^+ method. If we instead use smallpox parameters, we again see that low $\mathcal{R}_0$ values of 2 or 3 causes large estimation error, but the error is otherwise not very sensitive to $\mathcal{R}_0$ or α (see §S8 of the Supplementary Material). A sample estimate of the transmission rate in the presence of observation error is provided in §S8 of the Supplementary Material.

The true case fatality ratio (η) for measles is between 0.3% and 34% depending on the time period and location [45]. If we look at a range of $\rho\eta$ values from 0.1 to 1, the dependence of estimation error on α and $\mathcal{R}_0$ looks qualitatively identical to the $\rho\eta = 0.2$ case in Figure 6 (also see §S8 of the Supplementary Material). However, the magnitude of the estimation error increases as $\rho\eta$ decreases, as shown in Figure 7.

If we look at $\rho\eta < 0.1$, which is possible for a disease such as current day measles (case fatality for measles from 1987–2000 in the US was 0.3% [45]), our estimate of $\beta(t)$ is so noisy that is difficult to identify any characteristics of the true transmission rate (see §S8.5 of the Supplementary Material). Because the number of reported cases each week is too noisy a representation of the true disease dynamics, neither the S^+ or the SI method can estimate $\beta(t)$ well. If η is extremely small, than a useful estimate of $\beta(t)$ can be made only from incidence data, not mortality data.

4.4 Sensitivity to process error

The spread of an infectious disease in a population is a stochastic process. Also, in the deterministic SIR model, S, I, and R are taken to be continuous variables, instead of the whole

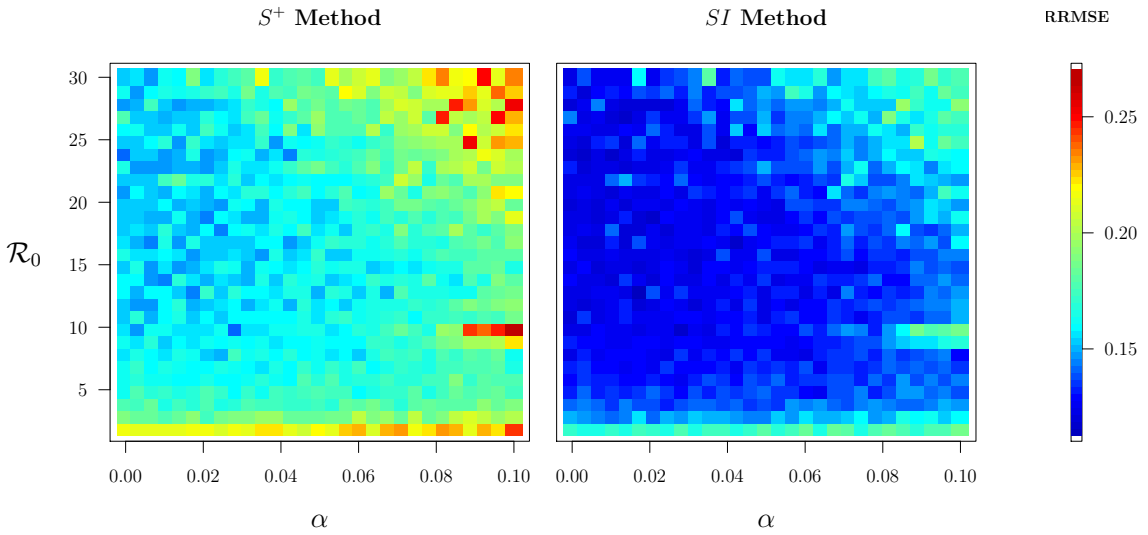


Figure 6: Sensitivity to observation error in estimating the transmission rate $\beta(t)$. The case notification data we observe in the world is noisier than the mock data simulated with the SIR model (22) because of observation error. In order to investigate this, we treat the recorded cases each time step Δt as a binomial process where the number of trials is the number of new infected cases in Δt and the probability of success is equal to the reporting/case fatality ratio ($\rho\eta$). Here $\rho\eta = 0.2$ and measles parameters are used (Table 2). For each α-$\mathcal{R}_0$ pairing where $\alpha \in [0, 0.1]$ and $\mathcal{R}_0 \in [2, 30]$, the relative root mean squared error (RRMSE) is computed for both estimation methods. In the presence of observation error it is clear that the SI method is much more accurate. Very small $\mathcal{R}_0$ values (such as 2 or 3) and large $\mathcal{R}_0$ and α values decrease estimation accuracy.

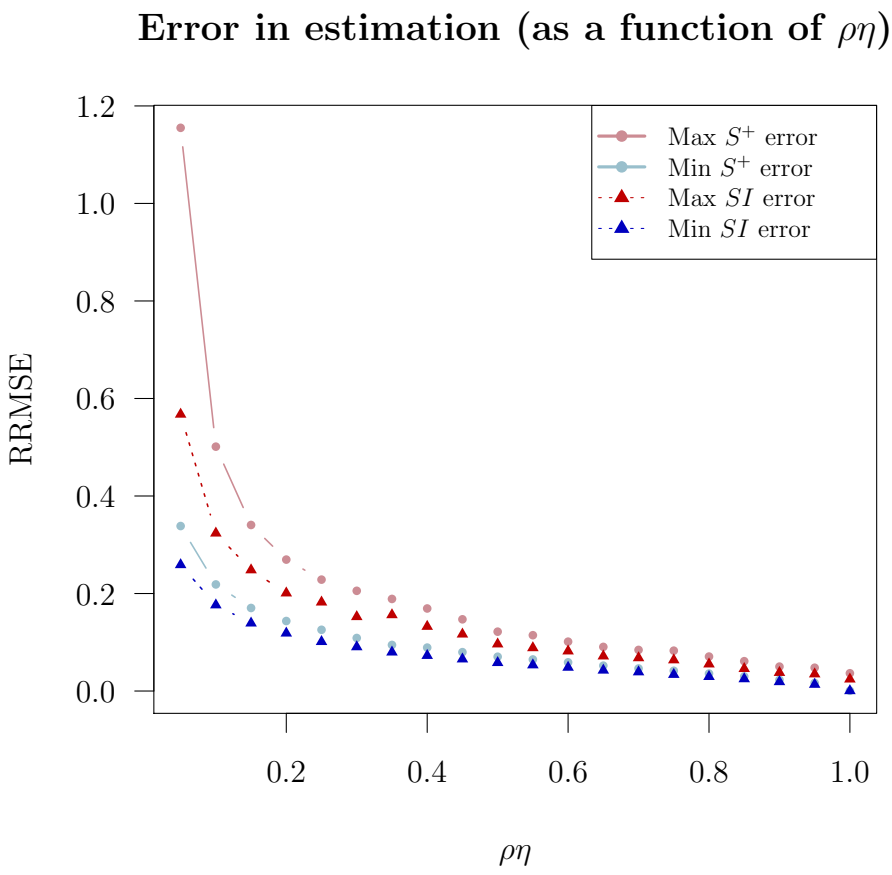


Figure 7: Maximum and minimum error in estimating the transmission rate $\beta(t)$ over a range of α and $\mathcal{R}_0$ values ($\mathcal{R}_0 \in [0, 30], \alpha \in [0, 0.1]$) if the simulated case notification data is sampled from a binomial distribution. Figure 6 shows how the estimates depend on the value of $\mathcal{R}_0$ and α for the reporting/case fatality ratio $\rho\eta = 0.2$. Varying $\rho\eta$ from 0.1 to 1 results in a qualitatively similar plot to Figure 6, except that the maximum and minimum error on the legend change. This figure shows the maximum and minimum error for both the S^+ and SI method as a function of $\rho\eta$.

numbers of individuals that we observe in the real world. We can incorporate discreteness and demographic stochasticity by using the Gillespie algorithm [23] to simulate case notification data. The Gillespie algorithm is a simple method that can provide realizations of the stochastic SIR model with a discrete population. We want to ensure that our estimation methods can predict $\beta(t)$ well in this more realistic situation.

With measles parameters, for each value of α and $\mathcal{R}_0$ we ran 100 realizations of the stochastic SEIR model using the Gillespie algorithm, and then used the case notification data from those realizations to estimate $\beta(t)$. Then for every α-$\mathcal{R}_0$ pairing, we took the median RRMSE in estimating $\beta(t)$ from the stochastic data for both the S^+ and SI methods. We examined population sizes of $N_0 = 100,000$, $500,000$, and $1,000,000$. Figure 8 shows the accuracy of each method's estimate of $\beta(t)$ if we use a population size of $100,000$. In this case, since the population size is so small, we often have fadeout of the disease before the end of the 20 years we are looking at. The far right panel of Figure 8 shows the probability of fade-out of the disease for each α-$\mathcal{R}_0$ pair. Smaller values of $\mathcal{R}_0$ and larger values of α contribute to a higher probability of fadeout, and greater error in estimation. With $\mathcal{R}_0 = 2$, the disease faded out before 20 years in almost every case. In general, there is larger error in estimation if we experience fadeout, because both methods have difficulty estimating $\beta(t)$ right before the disease fades out (see an example of this in §S9.3 of the Supplementary Material).

Figure 9 shows the RRMSE if we instead use the Gillespie algorithm with a population of $1,000,000$. In this case, as long as $\mathcal{R}_0 \geq 4$, we almost never have fadeout before 20 years, since the population is greater than the critical community size for measles [5, 6]. With this larger population size, we begin to see a similar pattern of error with respect to $\mathcal{R}_0$ and α as we did when observation error was added (Figure 6). For large $\mathcal{R}_0$ and large α there is greater error in estimation, as well as for small values of $\mathcal{R}_0$. With this larger population size, both the S^+ and SI methods predict $\beta(t)$ well, though the SI method is more accurate (the estimation error is $< 10\%$ of the mean value of $\beta(t)$ for the S^+ method and $< 8\%$ of

the mean value of $\beta(t)$ for the SI method). Sample $\beta(t)$-estimates for a population size of $100,000$, $500,000$, and $1,000,000$ in the presence of process error are presented in §S9 of the Supplementary Material.

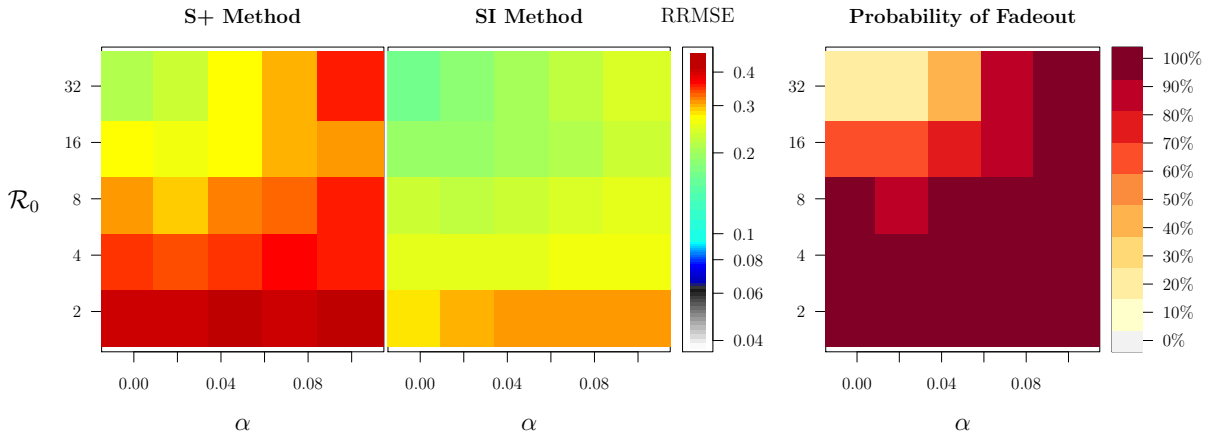


Figure 8: Sensitivity to process error when estimating the transmission rate $\beta(t)$ with a population of 100,000. For each α-$\mathcal{R}_0$ pairing (where $\mathcal{R}_0 \in \{2, 4, 8, 16, 32\}, \alpha \in \{0, 0.025, 0.05, 0.075, 0.01\}$), 100 Gillespie realizations were computed for measles parameters (Table 2) and used as mock case notification data. Then the S^+ and SI methods estimated $\beta(t)$ using each of the 100 simulated time series, and the median relative root mean square error (RRMSE) in estimation over all 100 data sets is recorded and plotted. Clearly the SI method is more accurate in estimating $\beta(t)$ for the range of $\mathcal{R}_0$ and α than the S^+ method. In the right panel, the probability of fadeout of the disease before 20 years is recorded. Since the population size is so small, if $\mathcal{R}_0 < 8$ the infectious disease always fades out before reaching 20 years. Estimation of $\beta(t)$ is especially difficult right before fadeout of the disease, which is why the estimation error is so high for small values of $\mathcal{R}_0$. Estimation error for both methods seem to correspond exactly to the probability of fadeout.

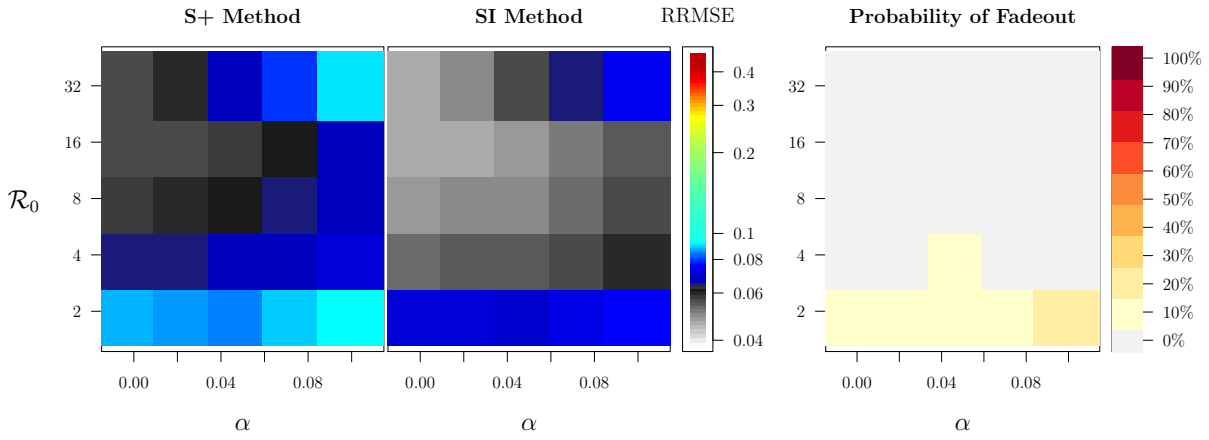


Figure 9: Sensitivity to process error when estimating the transmission rate $\beta(t)$ (as in Figure 8 but with a population of 1 million). Once again, the SI method is more accurate in estimation than the S^+ method. With this larger population size, estimation of $\beta(t)$ is very accurate, with a maximum RRMSE of 0.097 for the S^+ method and 0.079 for the SI method. The right panel shows the probability of fadeout of the disease before 20 years. Since the population size is large enough, the disease rarely experiences fadeout in less than 20 years.

5 Discussion and conclusions

The goal of this paper was to derive a fast and accurate method to estimate the time-varying transmission rate, $\beta(t)$, with the ultimate goal of estimating $\beta(t)$ from very long disease notification time series (*e.g.,* the London Bills of Mortality which contain weekly notifications since 1662). We have presented three methods for estimation of $\beta(t)$ given a time series of case notification data: the 'S method' as proposed by Fine and Clarkson [21], the 'S^+ method' as proposed by Krylova [37], and the 'SI method' proposed here. Each method was tested on simulated case notification data generated from the SIR model (Equation (1)) with a sinusoidally forced transmission rate $\beta(t)$, and we examined the methods' dependence on parameter values, and sensitivity to incorrect parameter values, observation error, and process error.

The S^+ method is an improved version of the S method, and performs much better in testing (*e.g.,* Figure 1). Krylova demonstrated that the S^+ method provided a qualitatively similar estimate for the amplitude of the seasonality of the transmission rate as Fine and Clarkson [21], Finkenstädt and Grenfell [22], and Hooker *et al.* [28] when testing it on the same weekly measles data [37]. We have found that the SI method presented in this paper performs even better than the S^+ method, especially in the case of a noisy transmission rate, or noise introduced by process or observation error.

If the mean generation time is short, or the initial state is far from the endemic equillibrium of the SIR model with a constant transmission rate (§3.1.2), there tends to be greater error in the estimation of $\beta(t)$. However, for a large range of parameter values (see Table 2), with $\beta(t)$ taken to be a sinusoidally forced function with amplitude $\alpha = 0.08$, the error in estimation is usually less than 4% of the mean value of $\beta(t)$ (see Figures 2–3). The transmission rate is easier to reconstruct accurately if its mean value and amplitude are small. The presence of noise in $\beta(t)$ makes estimation more difficult but the SI method still estimates $\beta(t)$ to within 4% (see Figure 4).

One major concern in estimating $\beta(t)$ with the fast methods presented here is that they

require a knowledge of disease parameter values. Often, one does not know the correct values for the population size, reporting ratio, *etc.,* of a data set. The fast estimation methods do not attempt to fit any parameter values and are especially sensitive to an underestimate of the birth rate $\nu(t)$, and the reporting/case fatality ratio $\rho\eta$ (Figure 5). In general, the need for independent parameter estimates is a significant limitation of the fast methods.

In order to use the SI method we need to know the mean generation time (γ^{-1}), the natural mortality rate $\mu(t)$, the number of births B_t, the time from infection until reporting T_{rep}, the reporting/case fatality ratio $\rho\eta$, and the initial number of susceptible and infected individuals in a population S_0 and I_0. For many infectious diseases we know the mean infectious and latent periods, so we can compute a reliable estimate for the mean generation time. In §4.2 we found that the SI method is not sensitive to a correct estimate of T_{rep}, and in §2 (Equation (21b)) we found that our estimate is not sensitive to I_0. For many data sets, vital statistics such as population size, the number of births, and the number of deaths are given. In that case we explicitly have birth data for B_t, and $\mu(t)$ can be computed as the number of deaths each time interval divided by the population size.

This leaves us with two parameters to estimate, the initial number of susceptibles S_0, and the reporting/case fatality ratio $\rho\eta$. An accurate fast method for estimating each of these parameters for a long data set would be extremely valuable. Krylova suggests picking S_0 so that the estimated number of susceptible individuals has no long-term trend for the first $5-10$ years [37]. One possible way to estimate $\rho\eta$ is to estimate the transmission rate using the SI method for a range of $\rho\eta$ values. Then we could use the estimated transmission rates and the SIR model (Equation (22)) to simulate case notification data, and see which $\rho\eta$ value generates a set of case notification data closest to the real data set.

We tested these fast estimation methods on binomially sampled simulated case notification data to mimic observation error (§4.3). The SI method accurately estimates $\beta(t)$ from these data sets as long as the case fatality/reporting ratio $\rho\eta$ is not too small (*i.e.,* $\rho\eta > 0.1$). As in the case without observation error, there is an increase in accuracy if $\mathcal{R}_0$ is small (but

larger than 3) and the amplitude of seasonality in $\beta(t)$ is small.

Lastly, sensitivity to process error was tested by applying the estimation methods to realizations of the stochastic SEIR model using the Gillespie algorithm (§4.4). For a population of 1 million, the SI method well approximates $\beta(t)$, with error less than 8% of the mean value of $\beta(t)$ (Figure 9). However in a smaller population size such as $100,000$, the infectious disease fades out quickly, which decreases the estimation accuracy of each method (Figure 8). This decrease in accuracy is presumably because the disease dynamics are dominated by demographic stochasticity rather than seasonal forcing in the transmission rate.

Difficulties arise when attempting to estimate the transmission rate from a poorly sampled epidemic (*i.e.,* with a very small reporting or case fatality ratio). For example, with measles parameters (Table 2) and a population size of $500,000$, if $\rho\eta = 0.01$ both the SI and S^+ method provide too noisy an estimate of $\beta(t)$ to identify any key characteristics of the transmission rate because very few cases are recorded in the case notification data. Of course, a data set with a low reporting/case fatality ratio presents a challenge for any method for reconstructing the transmission rate, not just the fast methods presented here.

In general, the SI method proposed in this paper estimates $\beta(t)$ accurately and quickly but relies on independent estimates of a number of parameters. It would be extremely beneficial to develop a fast method to fit parameters (such as the reporting ratio) to a long data set and to set up a statistical framework for the transmission rate estimation. Future work will also hopefully include analysis of the transmission rates for various diseases recorded in the London Bills of Mortality, which would provide a wealth of understanding of how infectious diseases change over time, and what factors influence transmission.

References

[1] S. Altizer, A. Dobson, P. Hosseini, P. Hudson, M. Pascual, and P. Rohani. Seasonality and the dynamics of infectious disease. *Ecology Letters*, 9:467–484, 2006.

[2] R. M. Anderson and R. M. May. *Infectious Diseases of Humans:Dynamics and Control*. Oxford University Press, Oxford, 1991.

[3] J. L. Aron and I. B. Schwartz. Seasonality and period-doubling bifurcations in an epidemic model. *Journal of Theoretical Biology*, 110:665–679, 1984.

[4] N. T. J. Bailey. *The mathematical theory of infectious diseases and its applications*. Hafner Press, New York, 1975.

[5] M. S. Bartlett. Measles periodicity and community size. *Journal of the Royal Statistical Society Series A*, 120:48–70, 1957.

[6] M. S. Bartlett. The critical community size for measles in the United States. *Journal of the Royal Statistical Society. Series A (General)*, pages 37–44, 1960.

[7] C. T. Bauch. The role of mathematical models in explaining recurrent outbreaks of infectious childhood disease. In F. Brauer, P. van den Driessche, and J. Wu, editors, *Mathematical Epidemiology*, volume 1945 of *Lecture notes in mathematics*, pages 297–319. Springer Berlin / Heidelberg, 2008.

[8] C. T. Bauch and D. J. D. Earn. Transients and attractors in epidemics. *Proceedings of the Royal Society of London, Series B*, 270(1524):1573–1578, 2003.

[9] D. Bernoulli. Essai d'une nouvelle analyse de la mortalité causée par la petite vérole et des advantages de l'inoculation pour la prévenir. *Mém Mathematical Physics Academy Royal Science Paris*, :1–45, 1766. (English translation entitled 'An attempt at a new analysis of the mortality caused by smallpox and of the advantages of inoculation to prevent it' In L. Bradley, *Smallpox Inoculation: An Eighteenth Century Mathematical Controversy.* Adult Education Department, Nottingham, 1971).

[10] D. Bernoulli and S. Blower. An attempt at a new analysis of the mortality caused by smallpox and of the advantages of inoculation to prevent it. *Reviews in Medical Virology*, 14(5):275–288, 2004.

[11] F. Brauer. Compartmental models in epidemiology. In F. Brauer, P. van den Driessche, and J. Wu, editors, *Mathematical Epidemiology*, volume 1945 of *Lecture notes in mathematics*, pages 19–79. Springer Berlin / Heidelberg, 2008.

[12] F. Brauer and C. Castillo-Chavez. *Mathematical models in population biology and epidemiology*, volume 40 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2001.

[13] K. Dietz. The incidence of infectious diseases under the influence of seasonal fluctuations. In *Mathematical Models in Medicine*, volume 11 of *Lecture Notes in Biomathematics*, pages 1 – 15. Springer-Verlag Berlin / Hiedelberg, 1976.

[14] S. F. Dowell. Seasonal variation in host susceptibility and cycles of certain infectious diseases. *Emerging infectious diseases*, 7(3):369, 2001.

[15] D. J. D. Earn. A light introduction to modelling recurrent epidemics. In F. Brauer, P. van den Driessche, and J. Wu, editors, *Mathematical Epidemiology*, volume 1945 of *Lecture Notes in Mathematics*, pages 3–17. Springer Berlin / Heidelberg, 2008.

[16] D. J. D. Earn. Mathematical epidemiology of infectious diseases. In M. A. Lewis, M. A. J. Chaplain, J. P. Keener, and P. K. Maini, editors, *Mathematical Biology*, volume 14 of *IAS Park City Mathematics Series*, pages 151–186. American Mathematical Society, 2009.

[17] D. J. D. Earn, D. He, M. B. Loeb, K. Fonseca, B. E. Lee, and J. Dushoff. Effects of school closure on incidence of pandemic influenza in Alberta, Canada. *Annals of Internal Medicine*, 156(3):173–181, 2012.

[18] D. J. D. Earn, P. Rohani, B. M. Bolker, and B. T. Grenfell. A simple model for complex dynamical transitions in epidemics. *Science*, 287(5453):667–670, 2000.

[19] F. Fenner, D. A. Henderson, I. Arita, Z. Jezek, and I. D. Ladnyi. *Smallpox and its Eradication*. Geneva: World Health Organization, 1988.

[20] P. E. M. Fine. The interval between successive cases of an infectious disease. *American Journal of Epidemiology*, 158(11):1039–1047, 2003.

[21] P. E. M. Fine and J. A. Clarkson. Measles in England and Wales — I: an analysis of factors underlying seasonal patterns. *International Journal of Epidemiology*, 11(1):5–14, 1982.

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

[23] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.

[24] D. He, J. Dushoff, T. Day, J. Ma, and D. J. D. Earn. Mechanistic modelling of the three waves of the 1918 influenza pandemic. *Theoretical Ecology*, 4(2):283–288, 2011.

[25] D. He and D. J. D. Earn. Epidemiological effects of seasonal oscillations in birth rates. *Theoretical Population Biology*, 72:274–291, 2007.

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

[27] H. W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, 2000.

[28] G. Hooker, S. P. Ellner, L. De Vargas Roditi, and D. J. D. Earn. Parameterizing state-space models for infectious disease dynamics by generalized profiling: measles in Ontario. *Journal of the Royal Society Interface*, 8(60):961–974, 2011.

[29] E. L. Ionides, C. Breto, and A. A. King. Inference for nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 103(49):18438–18443, 2006.

[30] M. J. Keeling and B. T. Grenfell. Understanding the persistence of measles: reconciling theory, simulation and observation. *Proceedings of the Royal Society of London Series B-Biological Sciences*, 269(1489):335–343, 2002.

[31] M. J. Keeling and P. Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, Princeton, New Jersey, 2008.

[32] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London Series A*, 115:700–721, 1927.

[33] W. O. Kermack and A. G. McKendrick. Contributions to the mathematical theory of epidemics: The problem of endemicity. *Proceedings of the Royal Society of London A*, 138(834):55–83, 1932.

[34] W. O. Kermack and A. G. McKendrick. Contributions to the mathematical theory of epidemics: Further studies of the problem of endemicity. *Proceedings of the Royal Society of London A*, 141(843):94–122, 1933.

[35] A. A. King, E. L. Ionides, C. Breto, S. P. Ellner, and B. E. Kendall. Statistical inference for partially observed Markov processes (R package). `http://pomp.rforge.r-project.org`, Version 0.25-4, 2009.

[36] A. A. King, E. L. Ionides, M. Pascual, and M. J. Bouma. Inapparent infections and cholera dynamics. *Nature*, 454(7206):877–879, 2008.

[37] O Krylova. *Predicting epidemiological transitions in infectious disease dynamics. Smallpox in historic London (1664 - 1930)*. PhD thesis, McMaster University, Canada, 2011. http://digitalcommons.mcmaster.ca/opendissertations/6214.

[38] O. Krylova and D. J. D. Earn. Effects of the infectious period distribution on predicted transitions in childhood disease dynamics. *Journal of the Royal Society Interface*, 10:20130098, 2013.

[39] J. Landers. *Death and the Metropolis: Studies in the Demographic History of London, 1670-1830*, volume 20 of *Cambridge Studies in Population, Economy and Society in Past Time*. Cambridge University Press, 1993.

[40] A. L. Lloyd. Destabilization of epidemic models with the inclusion of realistic distributions of infectious periods. *Proceedings of the Royal Society of London Series B-Biological Sciences*, 268(1470):985–993, 2001.

[41] W. London and J. A. Yorke. Recurrent outbreaks of measles, chickenpox and mumps. I. Seasonal variation in contact rates. *American Journal of Epidemiology*, 98(6):453–468, 1973.

[42] J. Ma and Z. Ma. Epidemic threshold conditions for seasonally forced SEIR models. *Mathematical Biosciences and Engineering*, 3(1):161–172, 2006.

[43] D. Mollison and S. Ud Din. Deterministic and stochastic models for the seasonal variability of measles transmission. *Mathematical Biosciences*, 117:144–177, 1993.

[44] L. F. Olsen and W. M. Schaffer. Chaos versus noisy periodicity: alternative hypotheses for childhood epidemics. *Science*, 249:499–504, 1990.

[45] R. T. Perry and N. A Halsley. The clinical significance of measles: a review. *The Journal of Infectious Disease*, 189 (Supplement 1):S4–S16, 2004.

[46] M. Pollicott, H. Wang, and H. Weiss. Extracting the time-dependent transmission rate from infection data via solution of an inverse ODE problem. *Journal of Biological Dynamics*, 6(2):509–523, 2012.

[47] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[48] D. Schenzle. An age-structured model of pre- and post-vaccination measles transmission. *IMA Journal of Mathematics Applied in Medicine and Biology*, 1:169–191, 1984.

[49] Karline Soetaert, Thomas Petzoldt, and R. Woodrow Setzer. Solving differential equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010.

[50] H. E. Soper. The interpretation of periodicity in disease prevalence. *Journal of the Royal Statistical Society*, 92:34–61, 1929.

[51] J. H. Tien, H. N. Poinar, D. N. Fisman, and D. J. D. Earn. Herald waves of cholera in nineteenth century London. *Journal of the Royal Society Interface*, 8(58):756–760, 2011.

[52] J. A. Yorke and W. London. Recurrent outbreaks of measles, chickenpox and mumps. II. Systematic differences in contact rates and stochastic effects. *American Journal of Epidemiology*, 98(6):468–482, 1973.

SUPPLEMENTARY MATERIAL
Fast estimation of time-varying transmission rates for infectious diseases

Michelle deJonge

May, 2014

Contents

S1 Introduction **42**

S2 Parameter Definition **42**
S2.1 Determining the initial conditions 42
S2.2 `R Code:` Defining measles and smallpox parameter sets 43
S2.2.1 `R Code:` Modifying parameter lists 45
S2.3 `R Code:` Constructing the sinusoidal transmission rate $\beta(t)$ 46

S3 `R Code:` Simulating case notification data **47**
S3.1 A sample time series generated by the SIR model 50

S4 Implementing the S, S^+, and SI Methods for estimating the transmission rate $\beta(t)$ **52**
S4.1 `R Code:` The S Method 52
S4.2 `R Code:` The S^+ Method 54
S4.3 `R Code:` The SI Method 56
S4.4 Choosing the left or right time endpoint of the observation interval to estimate $\beta(t)$ 58
S4.5 Solutions to the S_t and I_t recurrence relations 60
S4.5.1 A variable $\mu(t)$ 60
S4.5.2 A fixed μ 62
S4.5.3 Dependence on initial conditions 62

S5 Comparing the performance of the $\beta(t)$-estimation methods **63**
S5.1 `R Code:` Relative root mean square estimation error (RRMSE) 63
S5.2 `R Code:` Combining operations to easily estimate $\beta(t)$ 66
S5.3 Estimating the transmission rate: an example 67

S6 Dependence on parameter values **70**
S6.1 `R Code:` Dependence on $\gamma, \nu, \mu, S_0, \& I_0$ 71
S6.1.1 The dependence of estimation accuracy on parameter values with a 'noisy' $\beta(t)$ 81
S6.1.2 Why does the estimation error peak in value, when $\nu(t)$ is approximately twice as large as in the measles parameter set? 84
S6.2 `R Code:` Dependence on $\mathcal{R}_0$ and α 87
S6.2.1 Dependence on $\mathcal{R}_0$ and α with smallpox parameters 94
S6.2.2 Estimating transmission rates with large amplitude 95

S7 Sensitivity to incorrect parameters **100**
S7.1 `R Code:` Sensitivity to incorrect parameters 100
S7.2 Comparing sensitivity to incorrect parameter values between the S^+ and SI methods 103
S7.3 Sensitivity of incorrect parameters when estimating the transmission rate for 20 or 300 years 105

S8 Sensitivity to observation error **107**
S8.1 An example of estimating $\beta(t)$ with observation error 107
S8.2 Sensitivity to observation error in estimating $\beta(t)$ for smallpox parameters . 109
S8.3 Estimation error for a range of reporting/case fatality ratio values 110
S8.4 `R Code:` The dependence of estimation error on $\rho\eta$ 112
S8.4.1 Maximum and minimum $\beta(t)$-estimation error for $\rho\eta \in [0.05, 1]$ . . . 114
S8.5 Estimating the transmission rate with very small reporting/case fatality ratios 115

S9 Sensitivity to process error **117**
S9.1 `R Code:` Sensitivity to process error 118
S9.2 Comparing estimation error for a range of population sizes 125
S9.3 $\beta(t)$-estimation for an infectious disease that fades out 126
S9.4 Plotting the estimate $\beta(t)$ for a range of population sizes 128

S1 Introduction

This collection of supplementary material includes both additional figures that have not been included in the main text, and computer code that provides the computational details behind our results. The goal of this supplement is to provide enough details to the reader that our results are easily reproducible, and the presented estimation methods are easy to implement. This document has been written with the `knitr` [5] package, which allows easy integration of LaTeX documents and `R` code. All of the computations used for this document were conducted using `R version 3.1.0 (2014-04-10)` [4].

S2 Parameter Definition

S2.1 Determining the initial conditions

When testing the performance of the S^+ and SI methods for estimation of the transmission rate, we generate simulated case notification data from the SIR model (§3). Since the solutions of the sinusoidally forced SIR model oscillate around the equilibrium value of the unforced model, where $\beta(t) \equiv \beta_0$, the initial conditions for $S(t)$ and $I(t)$ are chosen to be the endemic equilibrium values of the unforced SIR model with constant vital dynamics, $\mu(t) = \nu(t) \equiv \mu$. We will derive these endemic equilibrium values here.

Consider the SIR model with an unforced transmission rate, $\beta(t) \equiv \beta_0$, and constant vital dynamics, $\mu(t) = \nu(t) \equiv \mu$, *i.e.,*

$$\frac{dS}{dt} = \mu N_0 - \beta_0 SI - \mu S\,, \tag{S1a}$$

$$\frac{dI}{dt} = \beta_0 SI - \gamma I - \mu I\,, \tag{S1b}$$

$$\frac{dR}{dt} = \gamma I - \mu R\,. \tag{S1c}$$

To find the endemic equilibrium values $(\hat{S}, \hat{I})$, of this system of equations set $\frac{dS}{dt} = 0$ and $\frac{dI}{dt} = 0$:

$$\frac{dS}{dt} = 0 \implies \mu N_0 - \beta_0 \hat{S}\hat{I} - \mu\hat{S} = 0, \tag{S2}$$

$$\frac{dI}{dt} = 0 \implies \beta_0 \hat{S}\hat{I} - \gamma\hat{I} - \mu\hat{I} = 0. \tag{S3}$$

Factoring Equation (S3), we see that

$$\beta_0\hat{S} - \gamma - \mu = 0 \qquad \text{OR} \qquad \hat{I} = 0. \tag{S4}$$

Since we are looking for the endemic equilibrium, we will discard the disease free equilibrium (when $\hat{I} = 0$). So we can rewrite the equilibrium number of susceptibles as

$$\beta_0\hat{S} - \gamma - \mu = 0 \implies \hat{S} = \frac{\gamma + \mu}{\beta_0}. \tag{S5}$$

Then using the definition of $\mathcal{R}_0$ as provided in Equation (25):

$$\mathcal{R}_0 = \frac{\beta_0 N_0}{\gamma + \mu}, \tag{S6}$$

we can rewrite $\hat{S}$ as

$$\hat{S} = \frac{N_0}{\mathcal{R}_0}. \tag{S7}$$

Then, we plug this value for $\hat{S}$ into Equation (S2) to find $\hat{I}$

$$\mu N_0 - \beta_0 \frac{N_0}{\mathcal{R}_0}\hat{I} - \mu \frac{N_0}{\mathcal{R}_0} = 0. \tag{S8}$$

Rearranging this equation for $\hat{I}$ we have

$$\hat{I} = N_0 \left(1 - \frac{1}{\mathcal{R}_0}\right) \frac{\mu}{\gamma + \mu}. \tag{S9}$$

Thus we choose the initial conditions for our simulated data sets to be

$$S_0 = \hat{S} \equiv \frac{N_0}{\mathcal{R}_0}, \tag{S10a}$$

$$I_0 = \hat{I} \equiv N_0 \left(1 - \frac{1}{\mathcal{R}_0}\right) \frac{\mu}{\gamma + \mu}. \tag{S10b}$$

S2.2 `R Code`: Defining measles and smallpox parameter sets

Throughout the main text we use two main sets of parameters, one for measles and one for smallpox (as defined in Table 2). The `R` function `param.define()` records the parameters (as in Table 2) and initial conditions (as in Equation (S10)), and outputs a parameter list.

```
param.define <- function(type = "Measles", ## type = "Measles" OR "Smallpox"
                         no.years = 20,  # number of years to look at
                         time.step = 1){ # data time step in weeks.

  ## Define parameters for both Measles and Smallpox cases:

  ## times we want SIR data at
  times <- seq(0, 52*no.years/time.step, by = 1)
  ## natural death rate
  yearly.mortality.rate <- 0.04 # per year
  mu <- (yearly.mortality.rate/52)*time.step # per time.step

  ## (1) Measles in London,
  if (type == "Measles"){
    ## size of the population at time t_0
```

```
  pop.size <- 500000
  ##In SEIR model mean generation time is the sum
  ## of the mean latent and mean infectious periods
  mean.gen.time <- 13 #days
  ## gamma.val = Rate of Recovery: Units = 1 / time.step
  gamma.val <- 7/mean.gen.time*time.step
  R0 <- 20 # Basic Reproductive Number
  ## time (in weeks) between infection and reporting
  t.report <- round(1/gamma.val)
  ## time (in weeks) between infection and recovery
  t.recover <- t.report
  ## Both t.report and t.recover must be rounded
  ## to the nearest Delta t (time between case
  ## notifications in the data set)
  ## Because they tell us how many weeks forward
  ## or backwards we need to look in the data set
  ## case fatality ratio * reporting ratio
  cf.RR <- 1
}

## (2) Smallpox in London, 1664 Parameters
else if (type == "Smallpox"){
  ## size of the population in London in 1664
  pop.size = 392400
  #in SEIR mode mean generation time is the sum of
  ## the mean latent and mean infectious periods
  mean.gen.time <- 22 # units = days.
  ## gamma.val = Rate of Recovery: Units = 1 / time.step
  gamma.val = 7/mean.gen.time*time.step
  R0 <- 4 # Basic Reproductive Number
  ## time (in weeks) between infection and reporting
  t.report <- round(1/gamma.val)
  ## time (in weeks) between infection and recovery
  t.recover <- t.report
  ## case fatality ratio * reporting ratio
  cf.RR <- 1
}

## otherwise print an error
else{
  stop("Parameter Type Not Recognized (must be = Measles OR Smallpox)")
}

## Set the initial conditions to be the endemic equillibrium
```

```
  ## values of the SIR model with an unforced transmission rate
  Init.S <- 1/R0
  Init.I <- (1 - 1/R0)*(mu)/(gamma.val + mu)
  Init.R <- 1 - Init.S - Init.I

  ## Births per week: (This is a vector so that
  ## when applying the estimation method to a data set that contains
  ## the number of births each time unit, we can easily incorporate
  ## the reported births by setting Birth.input to be the birth data.)
  Birth.input <- rep(round(pop.size*mu), length(times))

  ## output a parameter list that contains everything.
  param.list <- list(times = times,
                     pop.size = pop.size,
                     gamma.val = gamma.val,
                     mu = mu,
                     t.report = t.report,
                     t.recover = t.recover,
                     cf.RR = cf.RR,
                     R0 = R0,
                     Birth.input = Birth.input,
                     Init.S = Init.S,
                     Init.I = Init.I,
                     Init.R = Init.R)

  ## return the parameter list.
  return(param.list)
}
```

S2.2.1 `R Code`: Modifying parameter lists

Here we define a few functions that allow for easy modification of parameter lists. `replace.param()` replaces one parameter value in a list with a new value, `range.list()` creates a list of parameter sets, where each parameter set is the same except for one parameter that is varied across the list, and `extra.range.list()` takes in a list of parameter sets (as created by `range.list()`), and changes one parameter in each list according to the range specified (this allows us to vary two different parameters over a range of values in the same list of parameter sets).

```
replace.param <- function(param.set, param.index, new.val){
  param.set[[param.index]] <- new.val
  return(param.set)
}
```

```
range.list <- function(orig.param, # original parameter set
                       param.index, # the index of the parameter to vary
                       range){ # the range over which the parameter varies.
  ## create an empty list for space
  param.range.list <- vector('list', length(range))
  ## for every entry in the parameter range:
  for (index in 1:length(range)){
    ## change the original parameter in the original list to be the new value
    ## if range is just a normal vector:
    if (typeof(range) == "double"){
      orig.param[[param.index]] <- range[index]
    }
    ## else if range is a list (as in Birth data):
    else if(typeof(range) == "list"){
      orig.param[[param.index]]  <- range[[index]]
    }
    ## store this parameter set with the new value in the param.range.list
    param.range.list[[index]] <- orig.param
  }
  ## return the list of parameter ranges.
  return(param.range.list)
}

extra.range.list <- function(range.list, # list of parameter lists
                             param.index, # index over which we want to vary
                             range){ # range of varying parameter
  for (index in 1:length(range.list)){
    # if range is just a normal vector:
    if (typeof(range) == "double"){
      range.list[[index]][[param.index]] <- range[index]
    }
    ## Else if Range is a list (as in Birth data):
    else if(typeof(range) == "list"){
      range.list[[index]][[param.index]] <- range[[index]]
    }
  }
  return(range.list)
}
```

S2.3 R Code: Constructing the sinusoidal transmission rate $\beta(t)$

For a given set of parameter values we choose a transmission rate $\beta(t)$ to be used in generating the simulated case notification data. As in equations (24) and (26) of the main text, $\beta(t)$ is

defined to be

$$\beta(t) = \max\left\{\beta_0\left[1+\alpha\left(\cos\left(\frac{2\pi t}{Y}\right)+\epsilon\phi(t)\right)\right], 0\right\}, \tag{S11}$$

where

$$\beta_0 = \frac{\mathcal{R}_0(\gamma+\mu)}{N_0}. \tag{S12}$$

The constant Y is one year in the chosen time unit, $\phi(t)$ is a point drawn from a Normal$(0,1)$ distribution, and ϵ is the intensity of the noise term $\phi(t)$. In Equation (S11) we take the maximum of the sinusoid and zero to ensure $\beta(t)$ is never negative.

`Rand.Vec()` constructs the random vector $\phi(t)$ and saves the vector so that each $\beta(t)$ used in this document has the same random component.

```
Rand.Vec <- function(length){
  vec <- rnorm(length, mean = 0, sd = 1)
  write.csv(vec, "RandomVectorForBeta.csv", row.names = FALSE)
}
```

`Create.Beta()` constructs a discrete vector that represents the transmission rate $\beta(t)$, computed using Equation (S11).

```
Create.Beta <- function(param.list, amp, period, noise.percent){
  with(param.list,{
    ## read in the random vector computed by Rand.Vec
    rand.vec <- read.csv("RandomVectorForBeta.csv")[,1]
    ## compute the mean value of beta
    b0 <- R0*(gamma.val + mu)/pop.size # mean value of Beta
    ppy <- (365/7) # data points per year
    ## compute beta.vec
    beta.vec <- b0*(1 + amp*cos(2*pi*times/(ppy*period)) +
                      amp*noise.percent*rand.vec)
    ## check if there is any negative entries due to the noise term:
    neg.entries <- which(beta.vec < 0)
    ## replace these negative entries with zero
    beta.vec[neg.entries] <- 0
    ## return beta(t)
    return(beta.vec)
  })
}
```

S3 `R Code`: Simulating case notification data

For testing the $\beta(t)$-estimation methods, we simulate case notification data using a numerical solution of the SIR model. Let $Q(t)$ be the cumulative number of cases from the initial time

t_0 to t. We add an additional equation for $Q(t)$ to the SIR model, which will keep track of the cumulative number of new cases so far, *i.e.*,

$$\frac{dS}{dt} = \nu(t)N_0 - \beta(t)SI - \mu(t)S\,, \tag{S13a}$$
$$\frac{dI}{dt} = \beta(t)SI - \gamma I - \mu(t)I\,, \tag{S13b}$$
$$\frac{dR}{dt} = \gamma I - \mu(t)R\,, \tag{S13c}$$
$$\frac{dQ}{dt} = \beta(t)SI\,. \tag{S13d}$$

Recall that Δt is the observation interval, T_{rep} is the mean time from infection to reporting (rounded to the nearest Δt), and $\rho\eta$ is the reporting/case fatality ratio. If a fixed proportion $\rho\eta$ of cases are reported, the simulated case notification data (C_t) can be computed from the numerical solution of $Q(t)$ as follows:

$$C_{t-T_{\text{rep}}} = \rho\eta\Big(Q(t) - Q(t-\Delta t)\Big). \tag{S14}$$

The function `solve.SIR` numerically solves the SIRQ model in Equation (S13) given a set of parameters and the chosen $\beta(t)$, and uses this solution to simulate case notification data (as in Equation (S14)).

```
solve.SIR <- function(params, beta,
                      ## binom.dist == TRUE, means the case notification
                      ## data is taken from a binomial distribution of the
                      ## true incidence data.
                      binom.dist = FALSE){

  ## the deSolve package computes the solution to the SIR model
  library(deSolve)

  with(as.list(params), {

    ## a. Define functions that linearly interpolate Beta(t),
    ## and the births to find the appropriate
    ## value for each specific value of t when solving the SIRQ model
    Beta.lin <- approxfun(beta, method = "linear", rule = 2)
    Birth.lin <- approxfun(Birth.input, method = "linear", rule = 2)

    ## define the gradient function for the SIR model
    SIR <- function(time, state, parameters){
      with(as.list(c(state, parameters)), {
        ## find the correct Beta value for each time using Beta.lin
        Beta.point <- Beta.lin(time)
        Birth.point <- Birth.lin(time) # same with Birth.lin
```

```
      ## compute the derivatives.
    dS <- Birth.point - Beta.point*S*I - mu*S
    dI <- Beta.point*S*I - (gamma.val + mu)*I
    dR <- gamma.val*I - mu*R
    dQ <- Beta.point*S*I # cumulative number of reported cases
    ## return the list to de solve.
    return(list(c(dS, dI, dR, dQ)))
  })
}


## b. Solve the SIR model
## define the initial conditions and parameters.
init <- c(S = Init.S*pop.size, I = Init.I*pop.size,
          R = Init.R*pop.size, Q = 0)
parameters <- c(gamma.val = gamma.val, mu = mu, cf.RR)

## ode solves the system of equations
## We have to jump through a few hoops here to supress the messages
## outputted by the function ode
options(warn = -1) # ignore warnings (warn = -1)
# after we leave this function: restore default warnings (warn = 0)
on.exit(options(warn = 0))
## solve the SIR model
## capture.output doesn't allow ode to print any messages to
## our knitr file
capture.output(out <- ode(y = init, times = times, func = SIR,
                          parms = parameters, method = "lsoda",
                         maxsteps = 10000, atol = 1e-100))
output <- as.data.frame(out)

## c. Compute the Case Notifications
## output£Q is the sum of all cases up to point t
## Cases for time t will be just the cases from t - delta t, to t.
Cases <- output$Q[-c(1)] - output$Q[-c(length(output$Q))]
## Then move the Cases vector ahead t.report points so that it is
## reported t.report weeks in the future from when the infection started.
len <- length(Cases)
if (t.report != 0){
  Cases <- c(rep(NA, (t.report + 1)),
                          Cases[-(seq(len - t.report + 1, len))])}
else if (t.report == 0){
  Cases <- c(NA, Cases)
}
```

```
    ## Now we need to adjust the cases by the reporting ratio and
    ## case fatality ratio. we can either do this by drawing
    ## from a binomial distribution in binom.dist = TRUE
    if (binom.dist == TRUE){
       Cases <- rbinom(Cases, size = round(Cases),
                                          prob = cf.RR)
    }
    ## or we can simply multiply the cases at each point in time by the
    ## reporting ratio * case fataltity ratio.
    else{
      Cases <- cf.RR*Cases
    }

    ## In the case that the numerical ode solver ('lsoda') may have to quit
    ## early, fill in the rest of the time points with NA values
    len <- length(times)
    output.len <- dim(output)[1]

    if (output.len != len){
      difference <- len - output.len
      na.vec <- rep(NA, difference)
      na.dataframe <- data.frame(time = na.vec, S = na.vec,
                                 I = na.vec, R = na.vec, Q = na.vec)
      output <- rbind(output, na.dataframe)
      Cases <- c(Cases, na.vec)
    }

    ## return a data frame that has time, S, I, R, Beta, C, Births
    output <- data.frame(time = times, S = output$S,
                         I = output$I, R = output$R,
                         beta = beta,
                         C = Cases,
                         Births = Birth.input)

    return(output)
  } )
}
```

S3.1 A sample time series generated by the SIR model

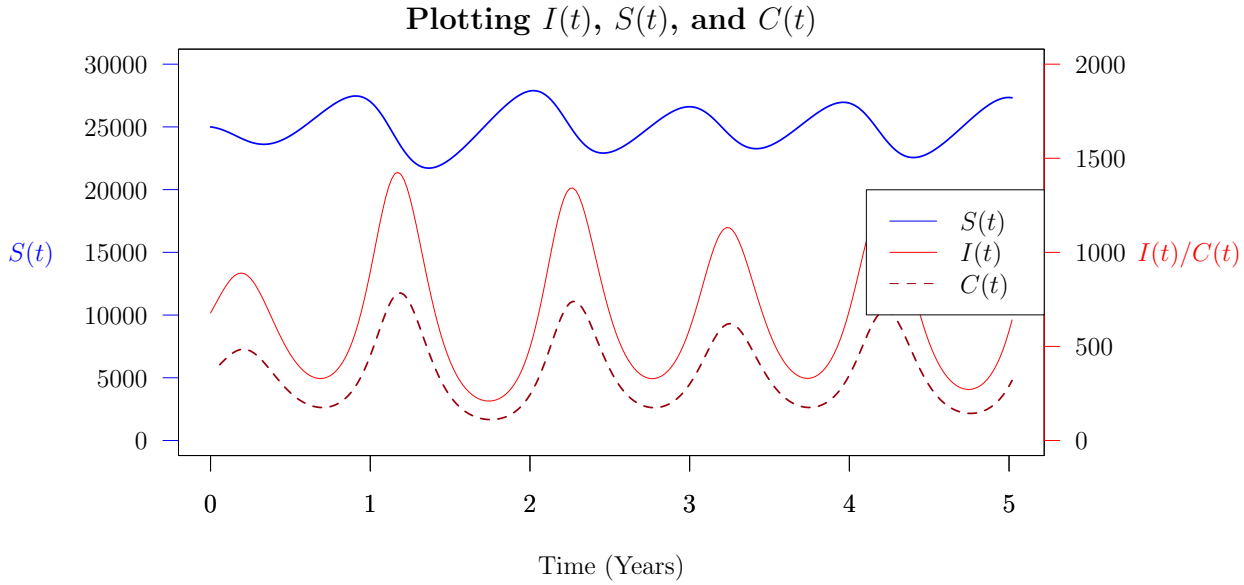
Let's look at a sample time series of susceptibles, infecteds, and simulated case notification data generated from a numerical solution to the SIR model. (In this case we use measles parameters (as in Table 2), and a sinusoidal $\beta(t)$ with seasonal amplitude $\alpha = 0.08$.)

```
## define parameters
param.meas <- param.define(type = "Measles", no.years = 20)
## define beta(t)
meas.beta <- Create.Beta(param.list = param.meas,
                         amp = 0.08, period = 1, noise.percent = 0)
## generate the SIR and case notification data
Data <- solve.SIR(params = param.meas, beta = meas.beta)
## Let's only look at the the first 5 years
end <- which(Data$time > 5*52)[1]
Data <- Data[1:end, ]
```

Print the first couple lines of the data set that is generated. The data set columns are *t*ime (in units of a week), S (the number of susceptibles), I (the number of infecteds), R (the number of removed individuals), *b*eta (the transmission rate at each point in time), C (the simulated case notifications), and *B*irths.

```
## Look at the first 10 lines of the dataset
print(Data[1:10,])

##    time     S     I      R      beta     C Births
## 1     0 25000 677.6 474322 2.329e-05    NA    385
## 2     1 24963 707.2 474330 2.328e-05    NA    385
## 3     2 24909 737.2 474354 2.324e-05    NA    385
## 4     3 24840 766.9 474394 2.318e-05 402.9    385
## 5     4 24756 795.3 474450 2.310e-05 419.4    385
## 6     5 24660 821.6 474521 2.299e-05 435.2    385
## 7     6 24553 844.8 474604 2.286e-05 449.7    385
## 8     7 24439 864.0 474699 2.272e-05 462.4    385
## 9     8 24321 878.3 474804 2.255e-05 472.7    385
## 10    9 24203 887.0 474914 2.238e-05 480.1    385
```

To look at this sample data set over a longer time period we plot the number of infected individuals ($I(t)$), the number of susceptible individuals ($S(t)$) and the number of case notifications ($C(t)$). The plotting code is suppressed. The vertical axis for $S(t)$ is on the left of the plot, and the vertical axis for $I(t)$ and $C(t)$ is on the right side of the plot.



S4 Implementing the S, S^+, and SI Methods for estimating the transmission rate $\beta(t)$

S4.1 `R Code:` The S Method

Fine & Clarkson presented a method to estimation the transmission rate of an infectious disease, which we refer to here as the 'S method'. The method estimates the transmission rate $\beta(t)$ with the following two equations.

$$S_{t+\Delta t} = S_t - Z_{t+\Delta t} + B_t, \tag{S15}$$

$$\beta(t) = \frac{Z_{t+\Delta t}}{Z_t S_t}. \tag{S16}$$

When applying the S method to a time series, the time between observations Δt is required to be equal to the mean generation time. The functions `agg.cases()` and `select.cases()` adjust the data set, so that the observation interval of the data is equal to the mean generation time. `agg.cases()` adds up the cases or births that have happened over the desired interval Δt, and `select.cases()` picks out the time point every Δt apart.

```
agg.cases <- function(vector, # vector to be aggregated
                      delta.t){
  ## delta.t is expressed as the number of points
  ## in the data set that we need to aggregate over.
  len <- length(vector)
  ## set up space for the new aggregated vector
  agg.vector <- rep(NA, round(len/delta.t))
```

```
  vector.index <- 1
  for(i in 1:length(agg.vector)){
    ## add up the cases/births/etc from over delta.t
    agg.vector[i] <- sum(vector[vector.index:
                                (vector.index + (delta.t - 1))],
                         na.rm = TRUE)
    vector.index <- vector.index + delta.t
  }
  return(agg.vector)
}


select.cases <- function(vector, delta.t){
  len <- length(vector)
  ## create space
  select.vec <- rep(NA, round(len/delta.t))
  vector.index <- 1
  for (i in 1:length(select.vec)){
    ## select the value every delta.t timepoints apart
    select.vec[i] <- vector[vector.index]
    vector.index <- vector.index + delta.t
  }
  return(select.vec)
}
```

The S method is implemented in `Est.FC()` which uses a set of estimation parameters and case notification data to estimate $\beta(t)$ and $S(t)$. Fine & Clarkson did not distinguish between exact incidence data Z_t and case notification data C_t, so we simply replace Z_t with C_t when applying the estimation method to case notification data.

```
Est.FC<- function(est.pars){
  with(est.pars,{
    ## a. aggregate the data to ensure Delta t = mean gen time
    ## mean.gen.time = mean generation time in weeks
    mean.gen.time <- round(1/gamma.val)
    ## add up the case notifications over the last Delta t time
    new.cases <- agg.cases(vector = C[1:length(C)], delta.t = mean.gen.time)
    ## add up the births over the last Delta t time
    agg.births <- agg.cases(vector = Birth.input,
                            delta.t = mean.gen.time)
    ## pick out the times Delta t apart
    times.out <- select.cases(vector = times, delta.t = mean.gen.time)

    ## b. Compute S(t)
```

```
    S <- rep(NA, length(new.cases)) # create space
    S[1] <- Init.S*pop.size # set initial condition
    for (k in 1:(length(S) - 1)){
      S[(k + 1)] <- S[k] + agg.births[k] -
        new.cases[(k + 1)]
    }

    ## c. Compute Beta(t).
    Beta <- rep(NA, length(new.cases)) # create space
    for (k in 1:(length(Beta)-1)){
      Beta[k] <- new.cases[(k + 1)]/(new.cases[k]*S[k]*mean.gen.time)
    }
    ## if Beta(t) is infinite, replace it with an NA
    Beta[which(is.infinite(Beta))] <- NA
    ## return the estimated S, Beta, and the times recorded.

    ## The second entry is set to be NA because the mortality
    ## data is delayed in such a way that the second entry would
    ## only have half the appropriate mortality entries
    Beta[2] <- NA

    return(list(S, Beta, times.out))
  })
}
```

S4.2 `R Code`: The S^+ Method

Krylova [3] modified the S method by deriving an estimate for the transmission rate direction from the SEIR model. We refer to her modified method as the 'S^+ method' which is defined by the following two equations.

$$\beta_t = \frac{1}{S_t}\frac{Z_t}{Z_{t-[T_{\text{inf}}]}}(\gamma + \mu_t), \tag{S17}$$

$$S_{t+\Delta t} = S_t + B_t - Z_{t+\Delta t} - \mu_t \Delta t S_t. \tag{S18}$$

In this document, the transmission rate is estimated from case notification data (C_t) as opposed to exact incidence data (Z_t). A proportion of the cases that occur over a time interval Δt are reported, and there is a delay between infection and reporting of each case. Thus we can relate C_t and Z_t as

$$C_{t+T_{\text{rep}}} = \rho\eta Z_t, \tag{S19}$$

where $\rho\eta$ is the reporting/case fatality ratio and T_{rep} is the mean time from infection to reporting. When applying the S^+ estimation method to case notification data we rewrite

the estimation method (in Equations (S17) & (S18)) in terms of C_t using Equation (S19).

$$\beta_t = \frac{1}{S_t}\frac{C_{t+T_{\text{rep}}}}{C_{t+T_{\text{rep}}-[T_{\text{inf}}]}}(\gamma + \mu_t), \tag{S20}$$

$$S_{t+\Delta t} = S_t + B_t - \frac{C_{t+T_{\text{rep}}+\Delta t}}{\rho\eta} - \mu_t \Delta t S_t. \tag{S21}$$

The S^+ method is implemented in `Est.S.plus()`, which uses a set of estimation parameters and case notification data to estimate $\beta(t)$ and $S(t)$.

```
## Est.S.plus: implementation of the S+ Method
Est.S.plus <- function(est.pars){
  with(est.pars,{

    ## a. Estimate S(t)
    S <- rep(NA, length(C)) # create space
    S[1] <- Init.S*pop.size # set initial condition
    ## compute S(t)
    for (k in 1:(length(S) - (1 + t.report))){
      S[(k + 1)] <- S[k] + Birth.input[k] -
        C[(k + 1 + t.report)]/cf.RR - mu*S[k]
    }

    ## b. Estimate Beta(t)
    Beta <- rep(NA, length(C)) # create space
    ## need to pick and start and end index because t.report
    ## pushes the case notification data forward.
    start.index <- 1 + t.recover - t.report
    end.index <- length(Beta) - t.report
    for(k in start.index:end.index){
      Beta[k] <- (1/S[k])*
        (C[(k + t.report)]/C[(k + t.report - t.recover)])*(gamma.val + mu)
      ## if Beta(t) is infinite, replace it with an NA
      Beta[which(is.infinite(Beta))] <- NA
    }

    ## return the estimated S and Beta
    return(list(S, Beta))
  })
}
```

S4.3 `R Code`: The SI Method

The SI method presented in this document estimates the transmission rate $\beta(t)$ by using a discrete-time approximation to the SIR model for both $S(t)$ and $I(t)$:

$$S_{t+\Delta t} = S_t + B_t - Z_{t+\Delta t} - \mu_t \Delta t S_t, \tag{S22}$$

$$I_{t+\Delta t} = I_t + Z_{t+\Delta t} - (\gamma + \mu_t)\Delta t I_t. \tag{S23}$$

Then $\beta(t)$ can be estimated in three different ways, depending on which t value we take (left endpoint, right endpoint, or average) for $\beta(t)S(t)I(t)$ over the observation interval Δt.

$$\beta_t = \frac{Z_t}{S_t I_t \Delta t} \qquad \text{(right endpoint)} \tag{S24a}$$

$$\beta_t = \frac{Z_{t+\Delta t}}{S_t I_t \Delta t} \qquad \text{(left endpoint)} \tag{S24b}$$

$$\beta_t = \frac{1}{2}\left[\frac{Z_t}{S_t I_t \Delta t} + \frac{Z_{t+\Delta t}}{S_t I_t \Delta t}\right]. \qquad \text{(average of endpoints)} \tag{S24c}$$

As in the discussion in §S4.2 about the S^+ method we want to apply the SI estimation method to case notification data (C_t) instead of exact incidence data (Z_t). Using the relationship

$$C_{t+T_{\text{rep}}} = \rho\eta Z_t \tag{S25}$$

the Z_t terms in the SI estimation method (Equations (S22), (S23) & (S24)) are rewritten in terms of C_t, *i.e.,*

$$S_{t+\Delta t} = S_t + B_t - \frac{C_{t+\Delta t+T_{\text{rep}}}}{\rho\eta} - \mu_t \Delta t S_t, \tag{S26}$$

$$I_{t+\Delta t} = I_t + \frac{C_{t+\Delta t+T_{\text{rep}}}}{\rho\eta} - (\gamma + \mu_t)\Delta t I_t. \tag{S27}$$

$$\beta_t = \frac{C_{t+T_{\text{rep}}}}{\rho\eta S_t I_t \Delta t} \qquad \text{(right endpoint)} \tag{S28a}$$

$$\beta_t = \frac{C_{t+\Delta t+T_{\text{rep}}}}{\rho\eta S_t I_t \Delta t} \qquad \text{(left endpoint)} \tag{S28b}$$

$$\beta_t = \frac{1}{2}\left[\frac{C_{t+T_{\text{rep}}}}{\rho\eta S_t I_t \Delta t} + \frac{C_{t+\Delta t+T_{\text{rep}}}}{\rho\eta S_t I_t \Delta t}\right]. \qquad \text{(average of endpoints)} \tag{S28c}$$

The SI method is implemented in `Est.SI()`, which uses a set of parameters and case notification data in order to estimate $S(t)$, $I(t)$ and $\beta(t)$. The SI method returns three estimates of $\beta(t)$ corresponding to the three different endpoints in equations (S28a).

```
Est.SI <- function(est.pars){
  with(est.pars, {

  ## a. Estimate S(t)
  S <- rep(NA, length(C))  ## make space
  ## set the initial condition for S.
  S[1] <- Init.S*pop.size
  for (k in 1:(length(S) - (1 + t.report))){
    S[(k + 1)] <- S[k] + Birth.input[k] -
      C[(k + 1 + t.report)]/cf.RR - mu*S[k]
  }

  ## b. Estimate I(t)
  I<- rep(NA, length(C)) ## make space
  ## set the initial condition
  I[1] <- Init.I*pop.size
  for (k in 1:(length(I) - (1 + t.report))) {
    I[(k + 1)] <- I[k] +
     C[(k + 1 + t.report)]/cf.RR - (gamma.val + mu)*I[k]
  }

  ## c. Estimate Beta(t) three ways:

  ### I. Right Boundary.
  Beta.1 <- rep(NA, length(C))
  for (k in 1:(length(Beta.1) - t.report)){
    Beta.1[k] <- C[(k + t.report)]/(S[k]*I[k]*cf.RR)
  }
  ## 2. Left Boundary
  Beta.2 <- rep(NA, length(C))
  for (k in 2:(length(Beta.2) - t.report)){
    Beta.2[(k-1)] <- C[(k + t.report)]/(S[(k-1)]*I[(k-1)]*cf.RR)
  }
  ## 3. Average of the Left and Right Boundary
  Beta.3 <- (Beta.1 + Beta.2)/2

  ## Return the estimate of S, I, and each Beta estimate.
  return(list(S, I, Beta.Av = Beta.3,
              Beta.End = Beta.1, Beta.Start = Beta.2))

  })

}
```

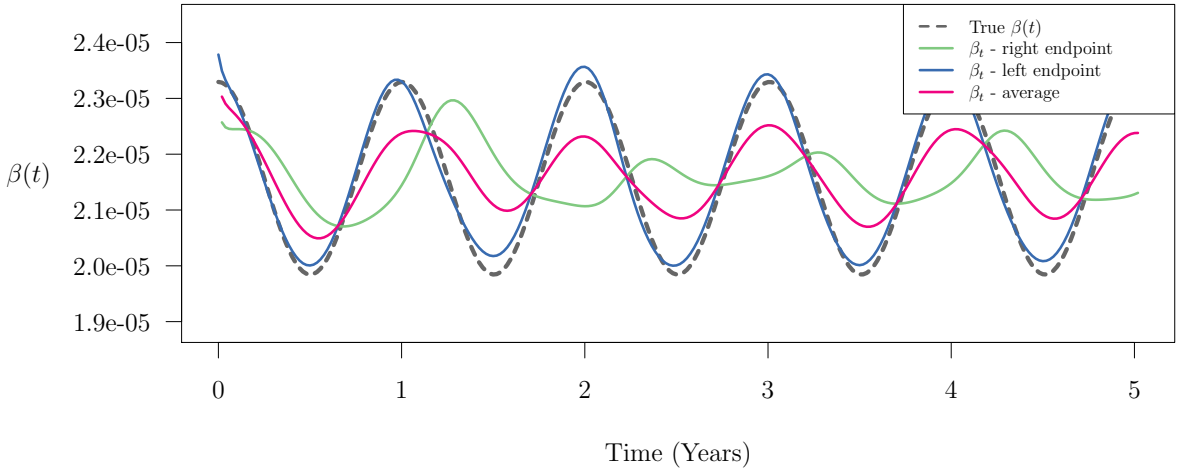
S4.4 Choosing the left or right time endpoint of the observation interval to estimate $\beta(t)$

In the derivation of both the S^+ and SI methods, it is necessary to make an assumption that over each observation interval Δt, $\beta(t)$, $S(t)$ and $I(t)$ are approximately constant. Then we assume $\beta(t)S(t)I(t) \approx \beta(\tau)S(\tau)I(\tau)$ over each Δt, where τ is some point in the time interval. Since we will only have estimates of $S(t)$ and $I(t)$ every Δt in time, our choice for τ is limited to the left or right endpoint of the time interval. Krylova [3] chose to use the right endpoint when deriving the S^+ method. In deriving the SI method, it was initially unclear which endpoint to pick, and if there would be any significant difference in estimation accuracy between the two choices. We found that choosing the left endpoint of each time interval produced a more accurate estimate for a wide range of transmission rates and parameter values.

In order to illustrate the resulting estimates for different endpoints, we plot a sample $\beta(t)$ along with its estimate using the left and right endpoints of the time interval. We also plotted the average of the two estimates of β_t. In the plot we can clearly see that using the left endpoint is much more accurate in estimating $\beta(t)$.

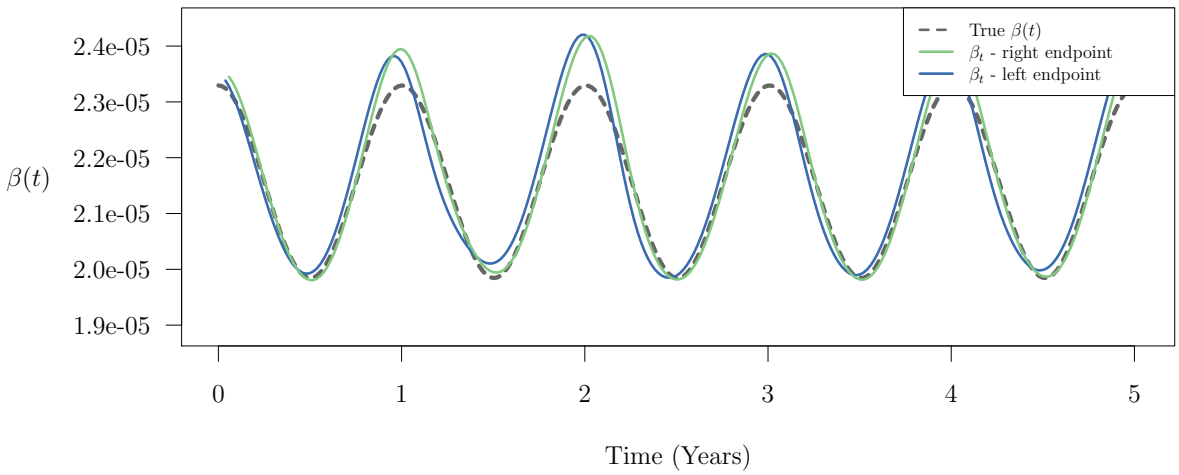
```
## define our parameter set (measles)
param.meas <- param.define(type = "Measles", no.years = 20)
## define beta(t)
meas.beta <- Create.Beta(param.list = param.meas,
                         amp = 0.08, period = 1, noise.percent = 0)
## generate case notification data
CN.Data <- solve.SIR(params = param.meas, beta = meas.beta)
## estimate beta(t) using the SI method with each endpoint
estimates <- Est.SI(est.pars = append(param.meas, list(C = CN.Data$C)))
est.left.endpoint <- estimates[[5]]
est.right.endpoint <- estimates[[4]]
est.av.endpoint <- estimates[[3]]
```

Estimating $\beta(t)$ With the Left or Right Endpoint (SI Method)



Seeing as the SI method does much better with the left time endpoint than the right, we revisited the derivation of the S^+ method to see if it would performs better with the left time endpoint as well (instead of the assumed right endpoint). We computed and evaluated estimates of $\beta(t)$ using the S^+ method with the left and right time endpoints. The sample plot shown below demonstrates that the estimate of $\beta(t)$ changes is very similar whether the left or right endpoint is used, and the estimate is actually more accurate for the right endpoint.

Estimating $\beta(t)$ With the Left or Right Endpoint (S^+ Method)



S4.5 Solutions to the S_t and I_t recurrence relations

S4.5.1 A variable $\mu(t)$

The SI method uses a discrete-time approximation to the continuous SIR model to estimate $S(t)$ and $I(t)$. These estimates are defined recursively where each value of S_t and I_t depends on the previous value in time. The discrete-time approximations are

$$S_{t+\Delta t} = S_t + B_t - Z_{t+\Delta t} - \mu_t \Delta t S_t, \tag{S29}$$

$$I_{t+\Delta t} = I_t + Z_{t+\Delta t} - (\gamma + \mu_t)\Delta t I_t. \tag{S30}$$

These recurrence relations can be solved to provide an explicit estimate for $S(t)$ and $I(t)$. Setting the initial time $t_0 = 0$ the solutions are

$$S_{j\Delta t} = S_0 \prod_{k=0}^{j-1}(1-\mu_{k\Delta t}\Delta t) + B_{(j-1)\Delta t} - Z_{j\Delta t} + \sum_{l=1}^{j-1}\prod_{k=l}^{j-1}(1-\mu_{k\Delta t}\Delta t)\left[B_{(l-1)\Delta t} - Z_{l\Delta t}\right] \qquad j = 1, 2, 3, .., \tag{S31}$$

$$I_{j\Delta t} = I_0 \prod_{k=0}^{j-1}(1-(\gamma+\mu_{k\Delta t})\Delta t) + Z_{j\Delta t} + \sum_{l=1}^{j-1}\prod_{k=l}^{j-1}(1-(\gamma+\mu_{k\Delta t})\Delta t)\, Z_{l\Delta t} \qquad j = 1, 2, 3, ... \tag{S32}$$

These explicit solutions can be inserted into the β_t estimate of the SI method (see Equation (S24)) to obtain an explicit expression of $\beta(t)$ for a time series Z_t.

The solutions to the recurrence relations can be proven using mathematical induction. We summarize and prove the solutions in Propositions 1 & 2.

Proposition 1. *If the initial time $t_0 \equiv 0$, then the solution to the recurrence relation*

$$I_{t+\Delta t} = I_t + Z_{t+\Delta t} - (\gamma + \mu_t)\Delta t I_t \tag{S33}$$

is

$$I_{j\Delta t} = I_0 \prod_{k=0}^{j-1}(1-(\gamma+\mu_{k\Delta t})\Delta t) + Z_{j\Delta t} + \sum_{l=1}^{j-1}\prod_{k=l}^{j-1}(1-(\gamma+\mu_{k\Delta t})\Delta t)\, Z_{l\Delta t} \qquad j = 1, 2, 3, ... \tag{S34}$$

Proof. (By Mathematical Induction)

Base Case: Verify the proposition holds true for $j = 1$ (*i.e.*, $t = 0$). Setting $t = 0$ in the recurrence relation (S33) results in

$$I_{\Delta t} = I_0 + Z_{\Delta t} - (\gamma + \mu_0)\Delta t I_0. \tag{S35}$$

Setting $j = 1$ in the proposed solution (S34) results in

$$I_{\Delta t} = I_0 \prod_{k=0}^{0}(1 - (\gamma + \mu_{k\Delta t})\Delta t) + Z_{\Delta t} \tag{S36a}$$

$$= I_0 + Z_{\Delta t} - (\gamma + \mu_0)\Delta t I_0\,. \tag{S36b}$$

So the proposition holds in the base case when $j = 1$ (*i.e.,* $t = 0$).

INDUCTION HYPOTHESIS: Assume that the proposition holds true for $j = (m-1), m > 1$, *i.e.,*

$$I_{(m-1)\Delta t} = I_0 \prod_{k=0}^{m-2}(1-(\gamma+\mu_{k\Delta t})\Delta t) + Z_{(m-1)\Delta t} + \sum_{l=1}^{m-2}\prod_{k=l}^{m-2}(1-(\gamma+\mu_{k\Delta t})\Delta t)Z_{l\Delta t} \tag{S37}$$

Now, prove that if the proposition holds true for $j = m-1, m > 1$, it also holds true for $j = m, m > 1$. By plugging $t = (m-1)\Delta t$ into the recurrence relation (S33) we gain an expression for $I_{m\Delta t}$, *i.e.,*

$$\begin{aligned} I_{m\Delta t} &= I_{(m-1)\Delta t} + Z_{m\Delta t} - (\gamma+\mu_{(m-1)\Delta t})\Delta t I_{(m-1)\Delta t} && \text{(S38a)}\\ &= (1-(\gamma+\mu_{(m-1)\Delta t})\Delta t)\left[I_{(m-1)\Delta t}\right] + Z_{m\Delta t}. && \text{(S38b)}\end{aligned}$$

Using the induction hypothesis, we can replace $I_{(m-1)\Delta t}$ in equation (S38b) with the expression for $I_{(m-1)\Delta t}$ in equation (S37), *i.e.,*

$$\begin{aligned} I_{m\Delta t} = (1-(\gamma+\mu_{(m-1)\Delta t})\Delta t)\Bigg[& I_0\prod_{k=0}^{m-2}(1-(\gamma+\mu_{k\Delta t})\Delta t) \\ & + Z_{(m-1)\Delta t} + \sum_{l=1}^{m-2}\prod_{k=l}^{m-2}(1-(\gamma+\mu_{k\Delta t})\Delta t)Z_{l\Delta t}\Bigg] + Z_{m\Delta t}. \end{aligned} \tag{S39}$$

Distributing the $(1-(\gamma+\mu_{(m-1)\Delta t})\Delta t)$ factor gives us:

$$I_{m\Delta t} = I_0 \prod_{k=0}^{m-1}(1-(\gamma+\mu_{k\Delta t})\Delta t) + Z_{m\Delta t} + \sum_{l=1}^{m-1}\prod_{k=l}^{m-1}(1-(\gamma+\mu_{k\Delta t})\Delta t)Z_{l\Delta t}. \tag{S40}$$

Thus this proposition holds true for $j = m$, so the proposition must hold true for all $j = 1, 2, 3, ...$ By mathematical induction, the solution to the recurrence relation for I_t is

$$I_{j\Delta t} = I_0\prod_{k=0}^{j-1}(1-(\gamma+\mu_{k\Delta t})\Delta t) + Z_{j\Delta t} + \sum_{l=1}^{j-1}\prod_{k=l}^{j-1}(1-(\gamma+\mu_{k\Delta t})\Delta t)\, Z_{l\Delta t} \qquad j = 1, 2, 3, ... \tag{S41}$$

□

Proposition 2. *If the initial time $t_0 \equiv 0$, then the solution to the recurrence relation*

$$S_{t+\Delta t} = S_t + B_t - Z_{t+\Delta t} - \mu_t \Delta t S_t \tag{S42}$$

is

$$S_{j\Delta t} = S_0\prod_{k=0}^{j-1}(1-\mu_{k\Delta t}\Delta t) + B_{(j-1)\Delta t} - Z_{j\Delta t} + \sum_{l=1}^{j-1}\prod_{k=l}^{j-1}(1-\mu_{k\Delta t}\Delta t)\left[B_{(l-1)\Delta t} - Z_{l\Delta t}\right] \qquad j = 1, 2, 3, ... \tag{S43}$$

Proof. Similar to proof of Proposition 1 □

S4.5.2 A fixed μ

If the data set we are interested in spans a short enough time period that the natural mortality rate μ_t is approximately constant, these recurrence relations become much simpler. With constant natural mortality the solutions to the recurrence relations stated in Equations (S31) & (S32) become

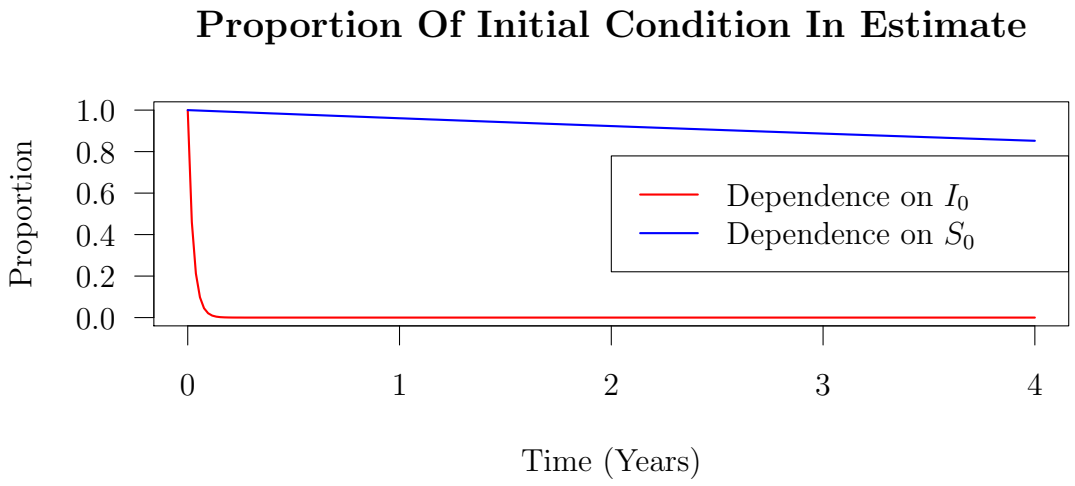
$$S_{j\Delta t} = S_0(1-\mu\Delta t)^j + \sum_{k=1}^{j}(1-\mu\Delta t)^{j-k}\left[B_{(k-1)\Delta t} - Z_{k\Delta t}\right] \qquad j = 1, 2, 3, .. \tag{S44}$$

$$I_{j\Delta t} = I_0(1-(\gamma+\mu)\Delta t)^j + \sum_{k=1}^{j}(1-(\gamma+\mu)\Delta t)^{j-k} Z_{k\Delta t} \qquad j = 1, 2, 3, ... \tag{S45}$$

S4.5.3 Dependence on initial conditions

The solutions to the recurrence relations display the dependence of the estimates S_t and I_t on the initial conditions S_0 and I_0. Here we will look at the case when the natural mortality rate μ is approximately constant over the data set (*i.e.*, we will use Equations (S44) & (S45)). In the estimates, the initial condition S_0 is multiplied by $(1-\mu\Delta t)^j$ and the initial condition I_0 is multiplied by $(1-(\gamma+\mu)\Delta t)^j$. For measles parameters(Table 2), and assuming $\Delta t = 1$ week, $(1-\mu\Delta t)^j \approx 0.999^j$, whereas $(1-(\gamma+\mu)\Delta t)^j \approx 0.461^j$. Thus as time increases, S_t and I_t will depend less and less on the initial conditions. However, dependence on I_0 decreases much more rapidly, than dependence on S_0. Plotting 0.461^j and 0.999^j as a function of time, we see that S_t depends on S_0 for a long time, whereas I_t rapidly has negligible dependence on I_0.

```
param.meas <- param.define(type = "Measles", no.years = 20)
with(param.meas,{
  weeks <- seq(0, 52*4, by = 1)
  ## assuming Delta t = 1 week,
  ## j in units of  1 week
  S0.depend <- (1 - mu)^weeks
  I0.depend <- (1 - (gamma.val + mu))^weeks
  plot(weeks/52, I0.depend, type = "l", col = "red", lwd = 2,
       main = "Proportion Of Initial Condition In Estimate",
       ylab = "Proportion", xlab = "Time (Years)", las = 1)
  lines(weeks/52, S0.depend, type = "l", col = "blue", lwd = 2)
  legend("right", c("Dependence on $I_0$", "Dependence on $S_0$"),
         col = c("red", "blue"), lwd = c(2,2), lty = c(1,1))
})
```



S5 Comparing the performance of the $\beta(t)$-estimation methods

S5.1 `R Code`: Relative root mean square estimation error (RRMSE)

In order to evaluate the performance of the $\beta(t)$-estimation methods, we compute the relative root mean square error between the true continuous $\beta(t)$ and the estimated discrete $\beta_{j\Delta t}$, $j = 1, 2, ...n$. The relative root mean squared error (RRMSE) is the euclidean distance between $\beta(t)$ and $\beta_{j\Delta t}$ divided by the square root of the number of points in $\beta_{j\Delta t}$, and the mean value of $\beta(t)$.

$$\text{RRMSE} = \sqrt{\frac{\sum_{j=1}^{n}\left(\beta(j\Delta t) - \beta_{j\Delta t}\right)^2}{n\left[\overline{\beta(t)}\right]^2}}, \tag{S46}$$

The `R` function `e.dist()` computes the euclidean distance between two vectors. If an element in one of the vectors is NA, the distance between the two vectors at that index is set to zero.

```
e.dist <- function(vec1, vec2){
  ## if the vectors arent the same length,
  if (length(vec1) != length(vec2)){
    # print an error
    print("Error in computing distance, vectors are of different length")
  }
  ## compute the distance
  summation <- 0
  for (index in 1:length(vec1)){
    ## as long as neither value vec1 or vec2 are NA
```

```
    ## add the distance between them to the summation
    if ((!is.na(vec1[index])) && (!is.na(vec2[index]))){
   summation <- summation + (vec1[index] - vec2[index])^2
    }
  }
  summation <- sqrt(summation)
  ## return the euclidean distance between the two vectors
  return(summation)
}
```

`error.est()` uses `e.dist()` to compute the RRMSE in estimating $\beta(t), S(t)$, and $I(t)$ with the $S+$ and SI methods.

```
error.est <- function(est.data, # estimated data
                      real.data, # true data
                      ## possible delay in computing error,
                      ## default is FALSE.
                      five.year.delay = FALSE){

  ## first check if five.year.delay is TRUE
  if (five.year.delay == TRUE){
    ## If so, cut off the first 5 years of the data set.
    start <- 5*52
    end <- dim(est.data)[1] - 52
    est.data <- est.data[start:end, ]
    real.data <- real.data[start:end, ]
  }

  ## compute the mean value of beta
  mean.val <- mean(real.data$beta)

  ## compute the number of points in the estimates that are not NA
  no.points <- dim(est.data)[1]
  no.NA.vals <- length(which(est.data$SI.S == NA))
  real.no.points <- no.points - no.NA.vals
  div <- sqrt(real.no.points)*mean.val

  ## compute error in S+ method
  S.plus.Beta.error <- e.dist(est.data$S.plus.Beta, real.data$beta)/(div)

  ## compute error in SI method (using left, right, average endpoints)
  SI.Beta.Av.error <- e.dist(est.data$SI.Beta.Av, real.data$beta)/(div)
  SI.Beta.End.error <- e.dist(est.data$SI.Beta.End, real.data$beta)/(div)
  SI.Beta.Start.error <- e.dist(est.data$SI.Beta.Start, real.data$beta)/(div)
```

```
  ## compute error in S_t, I_t estimates
  SI.S.error <- e.dist(est.data$SI.S, real.data$S)/(div)
  SI.I.error <- e.dist(est.data$SI.I, real.data$I)/(div)

  ## return the error in estimation
  output <- c(S.plus.Beta = S.plus.Beta.error,
              SI.S = SI.S.error, SI.I = SI.I.error,
              SI.Beta.Av = SI.Beta.Av.error,
              SI.Beta.End = SI.Beta.End.error,
              SI.Beta.Start = SI.Beta.Start.error)

  return(output)
}
```

When using the S method to estimate $\beta(t)$, we use a slightly different `R` function to compute error since the time interval between points (Δt) is adjusted to be equal to the mean generation time (as discussed in section §2).

```
## Error in estimating beta for the S method (Fine and Clarkson's method)
error.FC.est <-  function(est.beta, est.S,  # estimated beta and S
                          real.beta, real.S, # real beta and S
                          est.pars){ # parameter values
  with(est.pars, {

    ## First ensure that the timestep between two points of the
    ## true beta(t) and the true S(t) is equal to the
    ## mean generation time
    mean.gen.time <- round(1/(gamma.val)) # mean generation time in weeks
    delta.t <- mean.gen.time # our time step is the mean generation time
    ## select only the real Beta and S points every delta.t
    real.beta.dt <- select.cases(vector = real.beta,
                                 delta.t = mean.gen.time)
    real.S.dt <- select.cases(vector = real.S, delta.t = mean.gen.time)
    mean.val <- mean(real.beta)

    ## now since both the real and estimated beta and S
    ## have the same time distance between the points
    ## we can compute the error in estimation using e.dist
    no.points <- length(est.beta) #number of points
    no.NA.vals <- length(which(est.beta == NA)) # number of NAs
    real.no.points <- no.points - no.NA.vals # number of non-NA points
    div <- sqrt(real.no.points)*mean.val
    ## error in estimating S(t)
    S.error <- e.dist(real.S.dt, est.S)/div
```

```
    ## error in estimating beta(t)
    Beta.error <- e.dist(real.beta.dt, est.beta)/div

    output <- c(S = S.error, Beta = Beta.error)
    return(output)
  })
}
```

S5.2 `R Code`: Combining operations to easily estimate $\beta(t)$

In order to quickly generate simulated case notification data and estimate $\beta(t)$ with each of the methods, we define some additional functions that combine these operations. `Estimate.Beta()` estimates $S(t), I(t)$, and $\beta(t)$ using both the S^+ and SI methods given a set of parameters and case notification data.

```
## extended.params is a parameter set along with the generated case data
Estimate.Beta <- function(extended.params){
  ## Compute S and Beta using S+ method
  S.plus <- Est.S.plus(est.pars = extended.params)
  ## Compute S  I, and beta using the SI method
  S.SI <- Est.SI(est.pars = extended.params)
  ## Combine it into a data frame.
  output <- data.frame(S.plus.Beta = S.plus[[2]],  # Beta_t (S+ Method)
                       SI.S = S.SI[[1]],  # S_t
                       SI.I = S.SI[[2]], # I_t
                       SI.Beta.Av = S.SI[[3]], # Beta_t (SI Method - average)
                       SI.Beta.End = S.SI[[4]], # Beta_t (SI Method - right)
                       SI.Beta.Start = S.SI[[5]] #Beta_t (SI Method - left)
  )
  ## Then return the output
  return(output)
}
```

`Simple.comparison.NA()` generates the simulated case notification data, estimates $S(t)$, $I(t)$, and $\beta(t)$ using this data and the S^+ and SI method, and then computes the error in estimation. If more than 20% of the $\beta(t)$ estimate is NA, the estimation error is set equal to NA, so that we are not favouring estimates that just have more NA values. (NA values occur in the S^+ method if the case notification data is zero). `count.nas` is a helper function that computes the percent of the estimate that is NA.

```
Simple.comparison.NA <- function(beta, # true beta(t)
                                 params, # parameter values
                                 ## option to only count error after
```

```
                                  ## the first 5 years
                                  five.year.delay = FALSE){
  ## compute case notification data
  SIRdata <- solve.SIR(params = params, beta = beta)
  ## make an extended parameter list that also includes the
  ## case notification data
  extended.params <- append(params, list(C = SIRdata$C))
  ## Estimate Beta, S, and I using S+ and SI methods
  Estimates <- Estimate.Beta(extended.params)
  ## look at the percent of values in each estimate that
  ## is na, using the helper function count.nas
  na.percents <- count.nas(Estimates)
  ## compute the error in the estimation
  error <- error.est(est.data = Estimates,
                     real.data = SIRdata,
                     five.year.delay = five.year.delay)
  ## If more than 20\% of the values in an estimate are NA, we change
  ## the error term to be NA, so that we are aware of what is happening.
  for (index in 1:length(na.percents)){
    if (na.percents[index] >= 20){
      error[index] <- NA
    }
  }
  ## return the error vector
  return(error)
}

## count.nas is a helper function that counts
## the number of na values in the estimated data set
count.nas <- function(est.data){
  col.no <- dim(est.data)[2] # number of columns
  row.no <- dim(est.data)[1] # number of rows
  ## compute the percent of each column that is marked NA
  percent.na <- rep(NA, col.no)
  for (column in 1:col.no){
    percent.na[column] <- 100*length(which(is.na(est.data[,column])))/row.no
  }
  return(percent.na)
}
```

S5.3 Estimating the transmission rate: an example

With simulated case notification data generated from a sinusoidal $\beta(t)$ and measles parameters, we estimate the transmission rate using the S, S^+ and SI methods and measles

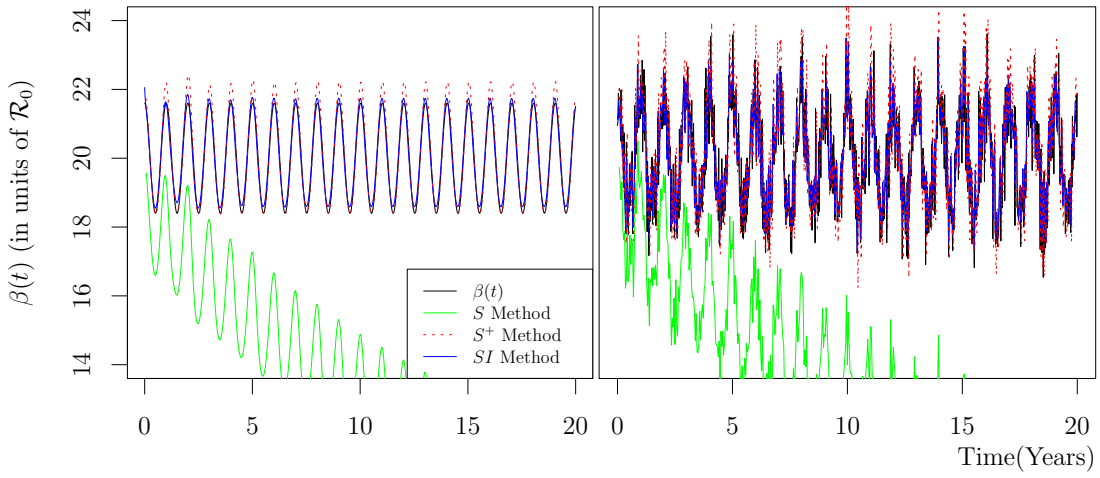
parameters (Table 2). The following serves as an example of implementing the previously defined functions in order to produce Figure 1 of the main text.

```
## First define the parameter values
param.meas <- param.define(type = "Measles", no.years = 20)

## Then define Beta(t)
## If there is not a random vector already created for Beta create one now
## check if the file "RandomVectorForBeta.csv" exists.
if(file.exists("RandomVectorForBeta.csv") == FALSE){
  ## if the file doesn't exist, call the function Rand.Vec
  # outputs a random vector with the same length as time.
  Rand.Vec(length(param.meas$time))
}
## we will have a beta(t) with epsilon = 0 (no noise)
meas.beta <- Create.Beta(param.list = param.meas, amp = 0.08,
                         period = 1, noise.percent = 0)
## and a beta(t) with epsilon = 0.5 (noise)
meas.beta2 <- Create.Beta(param.list = param.meas, amp = 0.08,
                          period = 1, noise.percent = 0.5)
```

```
## Compute the SIR data set, including the
## simulated case notification data
SIR.data <- solve.SIR(params = param.meas, beta = meas.beta)

## compute beta using the S Method (Fine & Clarkson)
FC.estimate <- Est.FC(est.pars = append(param.meas, list(C = SIR.data$C)))
FC.S <- FC.estimate[[1]]
FC.Beta <- FC.estimate[[2]]
FC.Time <- FC.estimate[[3]]
## compute the error in estimation.
FC.error <- error.FC.est(est.beta = FC.Beta, est.S = FC.S,
                         real.beta = meas.beta,
                         real.S = SIR.data$S,
                        est.pars = param.meas)

## compute beta using the S+ Method  and SI Method
Estimate <- Estimate.Beta(extended.params = append(param.meas,
                                                   list(C = SIR.data$C)))
SI.Beta <- Estimate[,6] # SI estimate
S.plus.Beta <- Estimate[,1] # S+ estimate

## compute the error in estimation
error <- error.est(est.data = Estimate,
```

```
                    real.data = SIR.data)

## Then let's plot each of the estimates of beta(t):
par(mfrow = c(1,2), oma = c(0,0,0,0)) # two panel plot
par(mar = c(3, 5, 0.1, 0.1)) # set up margins
with(param.meas,{
  yrs <- times/52 # time is in weeks, lets plot in years
  ## we want to plot beta in units of R0
  ## so need to multiply it by mult:
  mult <- pop.size/(gamma.val + mu)
  ## plot the real beta
  plot(yrs, meas.beta*mult, lwd = 1, ylim = c(14, 24), xlab = "",
       ylab = "$\\beta(t)$ (in units of $\\R$)", type = "l")
  ## plot S method estimate, has a different time step
  lines(FC.Time/52, FC.Beta*mult, col = "green", lty = 1, lwd = 0.8)
  ## plot SI method
  lines(yrs, SI.Beta*mult, col = "blue", lwd = 0.8)
  ## plot S+ method
  lines(yrs, S.plus.Beta*mult, col = "red", lty =3, lwd = 1)
  ## Create a legend
  legend("bottomright", c("$\\beta(t)$", "$S$ Method",
                          "$S^+$ Method", "$SI$ Method"),
         col = c("Black", "Green", "Red", "Blue"), lty = c(1, 1, 3, 1),
         lwd = c(1, 0.8, 0.4, 0.6), cex = 0.7)
  })

## Then, lets look at the case of a noisy beta(t).
##  (meas.beta2 defined above)
## Compute the SIR data with the noisy beta term,
## including the simulated case notificaiton data
SIR.data.N <- solve.SIR(params = param.meas, beta = meas.beta2)

## compute beta using the S Method
FC.estimate.N <- Est.FC(est.pars = append(param.meas,
                                          list(C = SIR.data.N$C)))
FC.S.N <- FC.estimate.N[[1]]
FC.Beta.N <- FC.estimate.N[[2]]
FC.Time.N <- FC.estimate.N[[3]]
## compute the error in estimation
FC.error.N <- error.FC.est(est.beta = FC.Beta.N, est.S = FC.S.N,
                           real.beta = meas.beta,
                           real.S = SIR.data.N$S, est.pars = param.meas)

## compute beta using the S+ Method  and SI Method
```

```
Estimate.N <- Estimate.Beta(extended.params = append(param.meas,
                                                     list(C = SIR.data.N$C)))
## compute the error in estimation
error.N <- error.est(est.data = Estimate.N,
                     real.data = SIR.data.N)
S.plus.Beta.N <- Estimate.N[,1] # S+ method
SI.Beta.N <- Estimate.N[,6] # SI method

## plot the results with a noisy beta.
par(mar = c(3, 0.1, 0.1, 4))

with(param.meas,{
  yrs <- times/52 # time is in weeks, lets plot in years
  mult <- pop.size/(gamma.val + mu)
  ## plot true beta
  plot(yrs, (meas.beta2*mult), lwd = 1, ylim = c(14, 24),
       xlab = "", ylab = "", type = "l", yaxt = "n")
  ## plot S method estimate
  lines(FC.Time.N/52, FC.Beta.N*mult, col = "green", lty = 1, lwd = 0.8)
  ## plot SI method estimate
  lines(yrs, SI.Beta.N*mult, col = "blue", lwd = 0.6)
  ## plot S+ method estimate
  lines(yrs, S.plus.Beta.N*mult, col = "red", lty = 3, lwd = 0.4)
  mtext("Time(Years)", adj = 1, side = 1, line = 2) # xaxis label
})
```



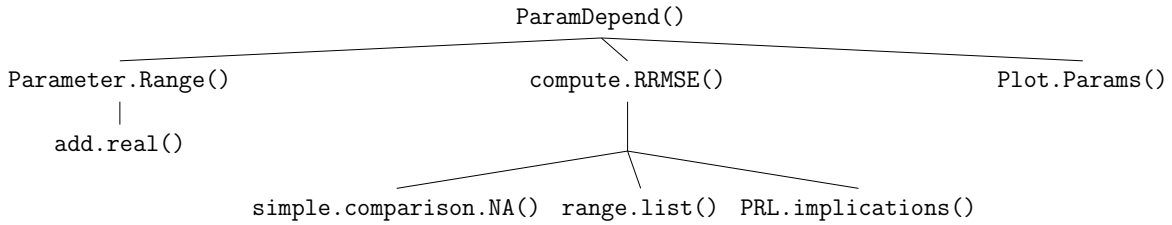
S6 Dependence on parameter values

It is important that the $\beta(t)$-estimation methods are accurate for a wide range of parameter values. To explore the parameter space, we start with a set of measles or smallpox parameters

(see Table 2) and then vary one parameter at a time by a factor of 4, from 25% to 400% of its value in the measles or smallpox parameter set (see Table 2).

S6.1 `R Code`: Dependence on $\gamma, \nu, \mu, S_0, \& \ I_0$

First, we explore the parameter space of the mean generation time γ^{-1}, the birth rate ν, the natural mortality rate μ, the initial amount of susceptibles S_0, and the initial amount of infected individuals I_0. We define several `R` functions to state the parameter range explored, compute the error in estimating $\beta(t)$ for each parameter in the range explored, and plot the resulting estimation error as a function of each parameter value. The structure of the functions used in this section is displayed in the following tree:



`ParamDepend()` is the overall function that is called when investigating dependence on parameter values. It uses `Parameter.Range()` to compute a vector that contains the range of values explored for the specified parameters, `compute.RRMSE()` to compute the error in estimating $\beta(t)$ for each value of the parameter in the range, and `Plot.Params()` to plot the resulting error for a variety of parameter values. We will begin with defining the `ParamDepend()` function.

```
ParamDepend <- function(beta, # chosen transmission rate
                        orig.param,  # original parameter set
                        ## option to compute the error after the first
                        ## five years have passed:
                        five.year.delay = FALSE,
                        method.name, # either "S+" or "SI.start"
                        min.error.val = 0, # min error for plotting
                        max.error.val,  # max error for plotting
                        max.percent, # max percent in param range
                        min.percent, # min percent in param range
                        stored.data.name, # so we can just reload the data
                        param.name, # specify for plotting
                        noise.percent = 0, # epsilon in beta definiton
                        right = FALSE,# if right=TRUE,yaxis drawn on the right
                        x.spot = 3.2, y.spot = 0.018, # legend location
                        legend = FALSE, # if legend = TRUE we plot a legend
                        cex.val = 1.5, # size of points.
                        ...){
```

```
## if the error in estimation as a function of param values has already
## been computed, then we load the saved error information from
## stored.data.name. If the estimation error has not yet been computed,
## stored.data.name will not exist, and we will go through the
## computation process.
if (!file.exists(stored.data.name)){
  ## The error as a function of a parameter value is stored in
  ## Error.Param.List.  Each entry corresponds to a data frame for
  ## a specific parameter (indicated by param.index)
  ## that contains the parameter range (as a % of the original value)
  ## and the error in estimating beta (t) for each value in that range.
  Error.Param.List <- vector(mode = "list", length = 6)
  ## EPL.i is the index of Error.Param.List
  EPM.i <- 1
  ## Length out is the number of points we want in our parameter range
  length.out <- 15
  ## index.4 checks if this is the first time we are looking at
  ## the natural mortality rate (= 0 if first time, else = 1)
  index.4 <- 0
  ## then for each parameter, compute the estimation error in the S+
  ## or SI methods for the range of explored values.
  for (param.index in c(3, 4, 4, 10, 11, 9)){

    ## first get the parameter range and percentange range
    ## (percentage range = parameter range / original parameter value)
    ## using the subfunction Parameter.Range
    output <- Parameter.Range(max.percent = max.percent,
                              min.percent = min.percent,
                              orig.param = orig.param,
                              param.index = param.index,
                              length.out = length.out)
    ## mu.only = TRUE means that only mu is varied, and nu stays equal to
    ## its value in the orig.param set.
    ## mu.only = FALSE, means that we vary nu alongside mu keeping
    ## the two parameters equal (so there is no change in pop size)
    ## if index.4 == 0, set mu.only = FALSE
    if (index.4 == 0){ mu.only <- FALSE }
    ## if index.4 == 1,set mu.only = TRUE
    else{ mu.only <- TRUE }
    ## if the parameter.index is 4, then increase the index.4 counter
    if (param.index == 4){ index.4 <- index.4 + 1 }

    ## Then for each value in the parameter range, compute
    ## the error in estimating beta (t) with the S+ or SI method.
```

```
      error.vec <- compute.RRMSE(orig.param = orig.param,
                                 param.index = param.index,
                                 range = output[[1]], # param range
                                 method.name = method.name, # S+ or SI.start
                                 five.year.delay = five.year.delay,
                                 mu.only = mu.only,
                                 noise.percent = noise.percent)
      ## store this estimation error into Error.Param.List as
      ## a data frame, with the first column equal to the percentage range
      ## and the second column equal to the error.
      Error.Param.List[[EPM.i]] <- data.frame(percent.range=output[[2]],
                                              error = error.vec)
      EPM.i <- EPM.i + 1
    }
    ## save the data set so we don't have to compute it again
    save(Error.Param.List, file = stored.data.name)
  }

  ## now lets plot the error
  ## define a list of colours
  col.list <- c("slateblue2", "darkseagreen4", "darkslategray3",
                "deeppink3", "mediumorchid2", "tan1")
   ## define the legend label names
  label.names <- c("$ \\gamma^{-1}$" , "$ \\mu = \\nu $",
                   "$ \\mu $","$ S_0 $", "$ I_0 $", "$ \\nu $")
  ## plot the data using the function Plot.Params
  Plot.Params(stored.data.name = stored.data.name,
              col.list = col.list,
              label.names = label.names,
              min.error.val = min.error.val,
              max.error.val = max.error.val,
              min.percent = min.percent, max.percent = max.percent,
              right = right,  x.spot = x.spot, y.spot = y.spot,
              cex.val = cex.val, param.name = param.name,
              legend = legend, ...)
}
```

Next we will define the `Parameter.Range()` function, which takes a parameter value specified in `param.index` and varies that parameter from 25% to 400% of its value in the measles or smallpox parameter set. In this section, we are defining `R` functions that help investigate how estimation accuracy depends on parameter values, but in the next section we will look at sensitivity of estimation accuracy to incorrect parameters values (§S7). Thus in the `Parameter.Range()` function, the marker `incorrect = TRUE` specifies that we are dealing with incorrect parameters.

```
## outputs a parameter range for the parameter in orig.param[param.index]
## where the range is from min.percent*param.val to max.percent*param.val
## spread equally across length.out points.
Parameter.Range <- function(max.percent, min.percent,
                            orig.param, # original parameter set
                            param.index,  # index of parameter to vary
                            length.out, # length of the parameter range
                            incorrect = FALSE){
  ## incorrect = TRUE => dealing with incorrect parameter values

  ## a. Find the original parameter value
  real.param <- orig.param[[param.index]]

  ## b. find the parameter range
  ## Case 1: Population Size (param.index == 2),
  ## mu(t) (param.index == 4)
  ## case fatality/reporting ratio (param.index == 7),
  ## S0 (param.index == 10), & I0 (param.index == 11)
  if (is.element(param.index, c(2, 4, 7, 10, 11))){
    ## vary the parameter from smallest to largest value
    range <- seq(min.percent*real.param, max.percent*real.param,
                 length.out = length.out)
    ## add.real adds the real.param to the range,
    ## if the marker incorrect = TRUE
    range <- add.real(range, real.val = real.param, incorrect)
    ## compute the percent range of the true parameter
    percentage.range <- range/real.param
  }
  ## Case 2. Mean Generation Time/ Gamma Val (param.index == 3)
  else if (param.index == 3){ # case 2
    ## In this case we actually want to vary the mean generation time
    ## not just gamma, and we increase/ decrease the mean gen time
    ## by one day incremets
    real.MGT <- 7/real.param # real mean generation time
    ## mean gen time range, increasing by 1 day each time
    MGT.range <- seq(round(min.percent*real.MGT),
                     round(real.MGT*max.percent), by = 1)
    ## our actual range is in terms of gamma:
    range <- 7/MGT.range
    percentage.range <- MGT.range/real.MGT
  }
  ## Case 3: T.report (param.index == 5), T.recover (param.index == 6)
  else if (param.index == 5 | param.index == 6){
    ## We need T.report and T.recover to be a multiple of Delta t
```

```
    ## Here the time unit is set to be equal to Delta t, so we need
    ## T.report and T.recover to be non-negative integers.
    range <- seq(round(min.percent*real.param),
                 round(max.percent*real.param), by = 1)
    percentage.range <- range/real.param
  }
  ## Case 4: Birth.input (param.index == 9)
  else if (param.index == 9){
    ## Birth.input is not just a parameter, but a vector
    ## of the number of births at each point in time.
    ## If we are varying birth.input, we first find nu(t)
    ## (the per capita birth rate) = births/population
    real.param <- orig.param[[9]][1]/orig.param[[2]]
    ## then find the range of nu(t).
    nu.range <- seq(min.percent*real.param, max.percent*real.param,
                    length.out = length.out)
    nu.range <- add.real(nu.range, real.val = real.param, incorrect)
    percentage.range <- nu.range/real.param
    ## then the range of birth.input is going to be a list of vectors
    range <- list(NA)
    for (index in 1:length(nu.range)){
      range[[index]] <- nu.range[index]*
        orig.param$pop.size*rep(1, length(orig.param[[9]]))
    }
  }
  ## if we don't fall under one of those cases, print a message
  else{
    print("Input for param.index is not in the range of accepted values.")
  }
  ## return the range of parameter values.
  return(list(range, percentage.range))
}
```

Parameter.Range() calls a sub-function add.real() that adds the true parameter value (from the base parameter set) to the parameter range if it is not in the range already. The 'true' value is only added if the marker incorrect = TRUE. Although this is only useful for the section dealing with sensitivity to incorrect parameters (§S7), since add.real() is a sub-function of Parameter.Range() we will define it here.

```
add.real <- function(range, # parameter range
                     real.val, # 'true' value of the parameter
                     incorrect){
  ## if we are looking at sensitivity to incorrect parameters,
  ## then incorrect = TRUE, and we want to add real.val to the range
```

```
  if (incorrect == TRUE){
    ## if real.val is NOT already in the range:
    if (!is.element(real.val, range)){
        ## then check if the range is increasing or decreasing:
        if ((range[2] - range[1]) < 0){
          decr.boolean <- TRUE
        }
        else{
          decr.boolean <- FALSE
        }
        ## and add the real.value accordingly.
        new.range <- sort(c(range, real.val), decreasing = decr.boolean)
    }
  }
  ## otherwise, real.val is in the range, so just return the original vector
  else{
    new.range <- range}
  return(new.range)
}
```

The main function `ParamDepend()` uses `compute.RRMSE()` to compute the error in estimating $\beta(t)$ for each parameter value in the parameter range given. `compute.RRMSE()` calls several sub-functions: `Create.Beta()` (defined in §S2.3), `Simple.comparison.NA` (defined in §S5.2), `range.list` (defined in §S2.2), and `PRL.implications()` (defined below).

```
compute.RRMSE <- function(orig.param, param.index,
                          range, method.name,
                          five.year.delay,
                          mu.only = FALSE, noise.percent = 0){
  ## for each element in the parameter range, create a parameter set
  ## that contains this element as the parameter value for param.index
  ## Then make a list of these parameter sets, one set for each
  ## element in the parameter range.
  param.range.list <- range.list(orig.param = orig.param,
                                 param.index = param.index, range = range)
  ## a change in one parameter will sometimes imply a change in another
  ## (eg. changing SO changes the other initial conditions)
  ## PRL.implications goes through and implements all the implied changes
  param.range.list <- PRL.implications(orig.param,
                                       param.range.list = param.range.list,
                                       param.index = param.index,
                                       mu.only = mu.only, range = range)
  ## then for every parameter set in param.range.list compute the
  ## error between the true beta (t) and the estimated beta_t
```

```
  index <- 1
  error.vec <- rep(NA, length(range)) # vector to store the estimation error
  for (param.set in param.range.list){
    ## compute beta (t) with the parameter values in param.set
    current.beta <- Create.Beta(param.list = param.set, amp = 0.08,
                                period = 1, noise.percent = noise.percent)
    ## compute estimation error
    error <- Simple.comparison.NA(beta = current.beta, params = param.set,
                                  five.year.delay = five.year.delay)
    if (method.name == "SI.start"){
      ## then estimation error is stored in the 6th element
      error.vec[index] <- error[6]
      index <- index + 1
    }
    else if (method.name == "S+"){
      ## then estimatino error is stored in the 1st element
      error.vec[index] <- error[1]
      index <- index + 1
    }
  }
  ## return the estimation error as a function of the parameter range.
  return(error.vec)
}
```

Many of our parameter values depend on other parameters. For example, if we change one of the initial conditions, the initial number of susceptible, infectious, and removed individuals must still add up to the initial population size. `PRL.implications()` ensures that a change in one parameter results in all dependent parameters being adjusted accordingly.

```
PRL.implications <- function(orig.param, param.range.list,
                             param.index, mu.only = FALSE, range = range,
                             incorrect = FALSE){
  ## Case 1: If gamma changes (ie mean gen time changes)
  if (param.index == 3){
    ## adjust T.report and T.recover to be the mean generation time rounded
    ## to the nearest observation interval.
    ## Change T.report (if incorrect = TRUE we don't change it because
    ## sensitivity to incorrect reporting time is something we later explore)
    if (incorrect == FALSE){
    param.range.list <- extra.range.list(range.list = param.range.list,
                                         param.index = 5,
                                         range = round(1/range))
    }
    ## Change the T.recover values
```

```
  param.range.list <- extra.range.list(range.list = param.range.list,
                                       param.index = 6,
                                       range = round(1/range))
}
## Case 2: If S0 or I0 changes
## If we are changing the initial conditions then we
## need to make sure they still all add up to one
## (they are stated as proportions of the population.)
else if (param.index == 10){ # if we changed S0
  ## Change the values of R.init to be 1 - I0 - S0
  I.init.val <- orig.param[[11]]
  param.range.list <- extra.range.list(range.list = param.range.list,
                                       param.index = 12,
                                       range = (1 - I.init.val - range))
}
else if (param.index == 11){ # if we changed I0
  # Change the values of R.init to be 1 - I0 - S0
  S.init.val <- orig.param[[10]]
  param.range.list <- extra.range.list(range.list = param.range.list,
                                       param.index = 12,
                                       range = (1 - S.init.val - range))
}
## Case 3: If the natural mortality rate is changing
else if (param.index == 4 & mu.only == FALSE){
  # if mu.only == FALSE we want to keep nu = mu when ranging mu
  pop.size <- orig.param$pop.size
  len <- length(orig.param$Birth.input)
  range.vec <- list(NA)
  for (index in 1:length(range)){
    range.vec[[index]] <- range[index]*pop.size*rep(1, len)
  }
  param.range.list <- extra.range.list(range.list = param.range.list,
                                       param.index = 9,
                                       range = range.vec)
}
## Case 4: If we change the population size,
## we need to adjust the recorded births
else if (param.index == 2){
  mu <- orig.param$mu
  len <- length(orig.param$Birth.input)
  range.vec <- list(NA)
  for (index in 1:length(range)){
    range.vec[[index]] <- range[index]*mu*rep(1, len)
  }
```

```
    param.range.list <- extra.range.list(range.list = param.range.list,
                                         param.index = 9,
                                         range = range.vec)
  }

  ## If the parameter that is ranging is NOT one of the initial conditions,
  ## We need to change the initial conditions so that we always start from
  ## the equillibrium value of the unforced model with constant beta
  param.range.list <- fix.IC(param.range.list, param.index, orig.param)
  return(param.range.list)
}
```

```
fix.IC <- function(param.range.list, # list of parameter sets
                   param.index, # index of the parameter that changed
                   orig.param){ # original parameter set
  ## if the parameter is mu, gamma, or population size:
  ## then adjust the initial conditions accordingly.
  if (is.element(param.index, c(2, 3, 4))){
    for (index in 1:length(param.range.list)){
      ## record the parameter values
      eGamma <- param.range.list[[index]]$gamma.val
      eMu <- param.range.list[[index]]$mu
      ePop <- param.range.list[[index]]$pop.size
      eR0 <- param.range.list[[index]]$R0

      ## Then define the new initial conditions.
      Init.S <- 1/eR0
      mean.beta <- eR0*(eGamma + eMu)/ePop
      Init.I <- (eR0 - 1)*eMu/(mean.beta*ePop)
      Init.R <- 1 - Init.S - Init.I

      param.range.list[[index]]$Init.S <- Init.S
      param.range.list[[index]]$Init.I <- Init.I
      param.range.list[[index]]$Init.R <- Init.R
    }
  }
  return(param.range.list)
}
```

The last function we need to define for this section is `Plot.Params()` which plots the relative root mean square error (RRMSE) in estimating $\beta(t)$ for the different parameter ranges.

```
Plot.Params <- function(stored.data.name, # estimation error
                        col.list, # list of plotting colours
                        label.names, # legend labels
                        min.error.val, # min error plotted
                        max.error.val, # max error plotted
                        right, # if right = TRUE, yaxis drawn on right side
                        min.percent,
                        max.percent, # max/min percent in param range
                        param.name, # parameter set name printed on plot
                        x.spot, y.spot, # legend location
                        cex.val = 1.5, # size of the points
                        legend = FALSE, ...){

  ## Start a plot that will show the error for a range of parameters.
  plot(0, 0, xlab = "Parameter Range", ylab = "",
       col = "white", ylim = c(min.error.val, max.error.val),
       xlim = c(min.percent, max.percent), ..., xaxt = "n", las = 1)

  ## decide on the x-axis labels of the plot:
  delta.x <- 0.5 # space between labels (50\%)
  num.vec <- 100*seq(0, max.percent, by = 0.5)
  ## Want to write the label as a percent not a proportion,
  ## so write out label names seperately
  label.vec <- rep(NA, length(num.vec))
  for(index in 1:length(label.vec)){
    label.vec[index] <- paste(num.vec[index], "\\%", sep = "")
  }
  ## Create the x-axis
  axis(side = 1, at = seq(0, max.percent, by = delta.x),
       labels = label.vec)
  ## If right = TRUE, create a yaxis on the right side of the plot
  if (right == TRUE){
    axis(4, las = 1)
  }
  else{ # if right = FALSE, label the yaxis with RRMSE
    mtext("RRMSE", side = 2, line = 4)
  }
  ## add a line showing where 100\% is.
  abline(v = 1, col = "gray39", lty = 2, lwd = 2)
  ## add gridlines to make the plot more readable.
  grid(lwd = 2, col = "gray80")

  ## Print the name of the parameter set on the plot
  ## in the top left corner (eg. Measles Parameters)
```

```
  if (max.percent == 4){ # if the max percent range is 4
    ## print the name at x = 3.5
  text(3.50, max.error.val, param.name, cex = 1.2)
  }
  else{ ## otherwise print the name at 1.75
    text(1.75, max.error.val, param.name, cex = 1.2)}

  ## Then for each dataframe in Error.Param.List plot the estimation error
  ## as a function of the percentage parameter range.
  load(stored.data.name) # contains Error.Param.List
  for (index in 1:length(Error.Param.List)){
    dataset <- Error.Param.List[[index]] # data set for one parameters
    ## plot error as a function of the percentage param range.
    lines(dataset[[1]], dataset[[2]], pch = 21,
          lwd = 4, type = "b", bg = col.list[index],
          col = col.list[index], cex = cex.val)
    ## outline the points in black
    points(dataset[[1]], dataset[[2]], pch = 1,
           col = "black", cex = cex.val)
  }

  ## the if legend == TRUE add a legend to the plot
  if (legend == TRUE){
    k <- length(label.names)
    legend(x = x.spot, y = y.spot, label.names,
           col = col.list, lty = rep(1, k), pch = rep(1,k),
           pt.cex = rep(cex.val, k), lwd = rep(4, k),
           bg = "white", cex = 1.2, text.width = 0.5)
  }
}
```

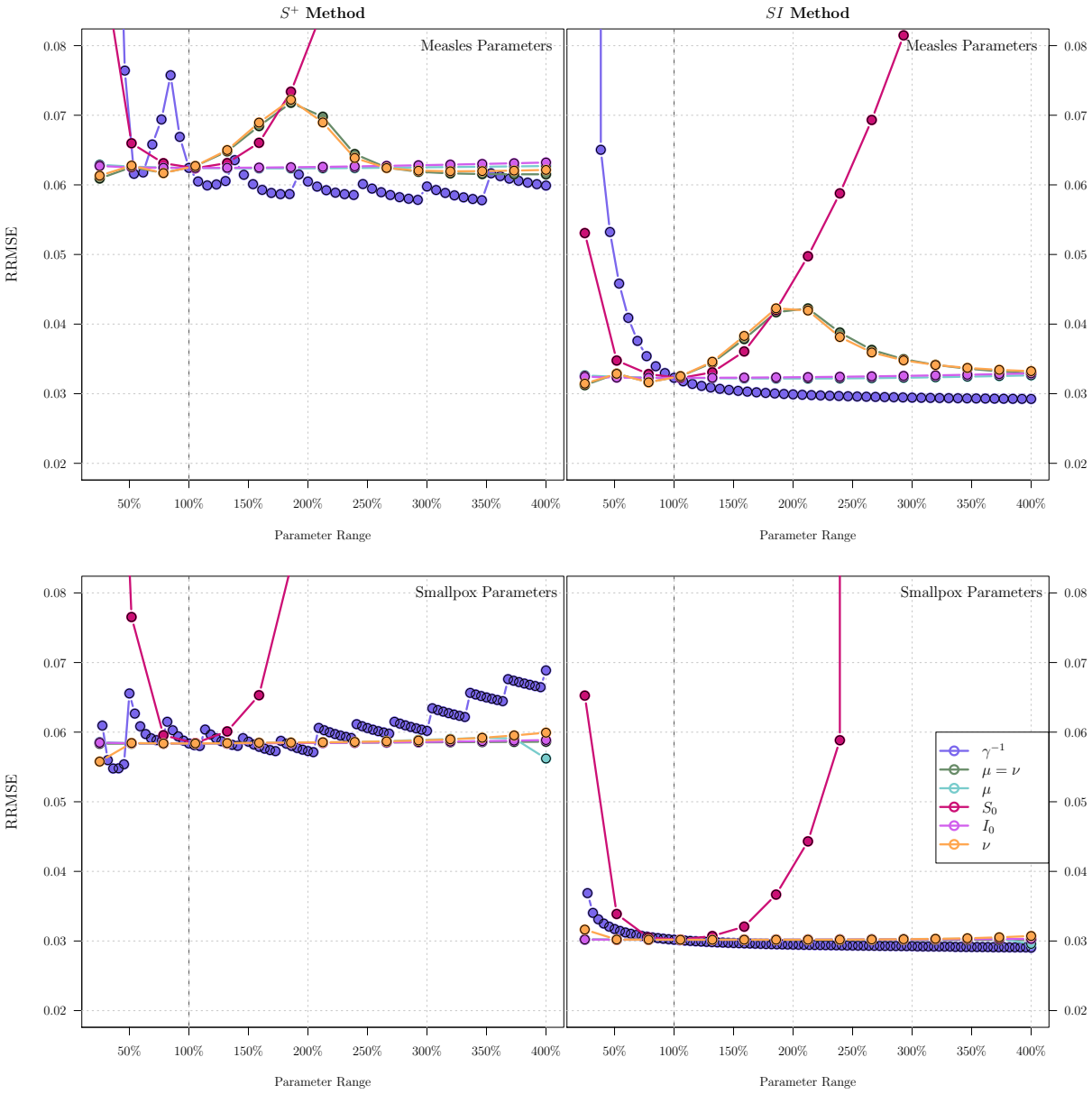
S6.1.1 The dependence of estimation accuracy on parameter values with a 'noisy' $\beta(t)$

In the main text, dependence of estimation accuracy for the SI and S^+ methods on parameter values was shown in Figure 2 for both measles and smallpox base parameter sets. There, we used a smooth $\beta(t)$ ($\epsilon = 0$ in Equation (S11)) to generate the simulated data and later be estimated by the S^+ and SI methods. Here we will produce a similar error dependence plot but for a 'noisy' $\beta(t)$, where $\epsilon = 0.5$ in Equation (S11).

```
## 1. Define the Parameters:
years <- 20
param.meas <- param.define(type = "Measles", no.years = years) # measles
param.small <- param.define(type = "Smallpox", no.years = years) # smallpox
```

```
## 2. Define Beta(t)
## If there is not a random vector already created for Beta create one now
## check if the file "RandomVectorForBeta.csv" exists.
if(file.exists("RandomVectorForBeta.csv") == FALSE){
  ## if the file doesn't exist, call the function Rand.Vec
  # which outputs a random vector for the time series
  Rand.Vec(length(param.meas$time))
}
## beta with measles parameters and epsilon = 0.5
meas.beta2 <- Create.Beta(param.list = param.meas,
                          amp = 0.08, period = 1, noise.percent = 0.5)
## beta with smallpox parameters and epsilon = 0.5
small.beta2 <- Create.Beta(param.list = param.small,
                           amp = 0.08, period = 1, noise.percent = 0.5)

## 3. Compute the estimation error as a function of parameter values
par(mfrow = c(2,2))
lb <- 0.02 # yaxis lower bound
ub <- 0.08 # yaxis upper bound

## Case 1: Measles, S+ Method
par(mar = c(5, 5, 2, 0) + 0.1)
ParamDepend(beta = meas.beta2, orig.param = param.meas,
            method.name = "S+",  min.error.val = lb, max.error.val = ub,
            max.percent = 4, min.percent = 0.25, five.year.delay = FALSE,
            stored.data.name = "SplusMeasles-Noisy.Rdata",
            main = "$S^+$ Method", param.name = "Measles Parameters",
            noise.percent = 0.5)

## Case 2: Measles, SI Method
par(mar = c(5, 0, 2, 5) + 0.1)
ParamDepend(beta = meas.beta2, orig.param = param.meas,
            method.name = "SI.start", min.error.val = lb, max.error.val = ub,
            max.percent = 4, min.percent = 0.25, five.year.delay = FALSE,
            stored.data.name = "SIMeasles-Noisy.Rdata",
            main = "$SI$ Method",  param.name = "Measles Parameters",
            noise.percent = 0.5, yaxt = "n", right = TRUE)

## Case 3: Smallpox, S+ Method
par(mar = c(6, 5, 1, 0) + 0.1)
ParamDepend(beta = small.beta2, orig.param = param.small,
            method.name = "S+", min.error.val = lb, max.error.val = ub,
            max.percent = 4, min.percent = 0.25, five.year.delay = FALSE,
            stored.data.name = "S+SmallPox-Noisy.Rdata",
```

```
            param.name = "Smallpox Parameters",
            noise.percent = 0.5)


## Case 4: Smallpox, SI Method
par(mar = c(6, 0, 1, 5) + 0.1)
ParamDepend(beta = small.beta2, orig.param = param.small,
            method.name = "SI.start", min.error.val = lb, max.error.val = ub,
            max.percent = 4, min.percent = 0.25, five.year.delay = FALSE,
            stored.data.name = "SISmallPox-Noisy.Rdata",
            param.name = "Smallpox Parameters", noise.percent = 0.5,
            yaxt = "n", right = TRUE, legend = TRUE, y.spot = 0.06)
```



The plot demonstrates the affect that different parameter values have on estimation error with a 'noisy' $\beta(t)$. With a noisy transmission rate the benefit of using the SI method over the S^+ method becomes clear. The SI estimation error is approximately twice as small than the S^+ estimation error. These plots bear a strong resemblance to Figure 2, except for an overall increase in error due to the noise in transmission. For both smooth and noisy transmission rates, estimation error is primarily influenced by S_0.

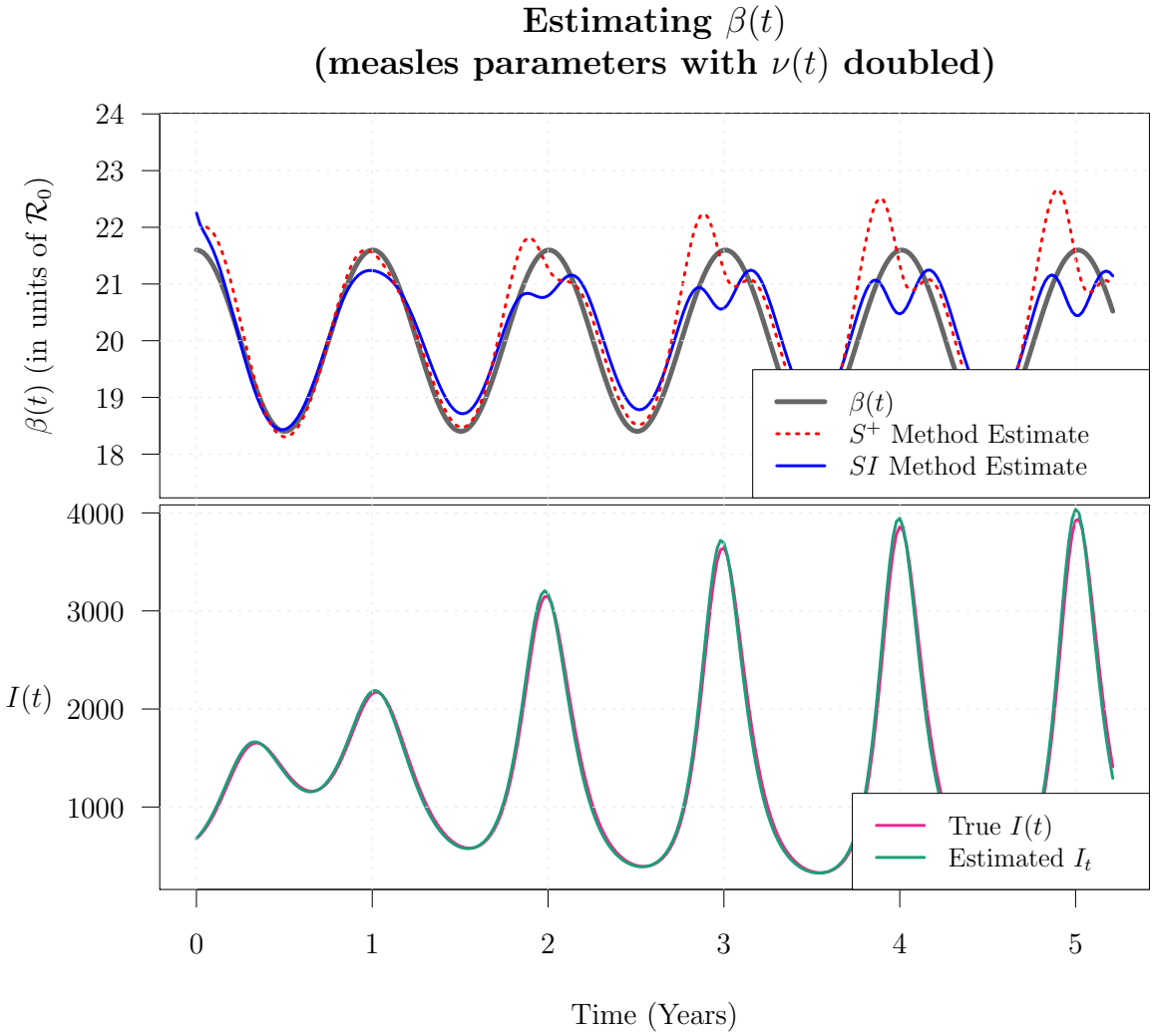
S6.1.2 Why does the estimation error peak in value, when $\nu(t)$ is approximately twice as large as in the measles parameter set?

When looking at dependence of estimation accuracy on parameter values, we found that for both a smooth and noisy transmission rate, error in estimation as a function of the birth

rate ν peaks if ν is twice as large as its value in the measles parameter set. In order to understand why this happens we plot the true $\beta(t)$ along with the estimated $\beta(t)$ in this case. Both the S^+ and SI methods estimate $\beta(t)$ well except for a dip in the estimates, at the peak of $\beta(t)$. These dips in estimation do not occur for the regular measles parameter set (as we saw in Figure 1). This dip is due to an overestimation of $I(t)$ at each of the points of maximum transmission. We can see this by looking at a plot of the estimated $\beta(t)$ and $I(t)$.

```
## start out with measles parameters
param.dip <- param.meas
## then double the amount of births
param.dip$Birth.input <- 2*param.dip$Birth.input
## then lets simulate case notification data and estimate beta(t)
## with each of our methods.
D1 <- solve.SIR(params = param.dip, beta = meas.beta)

## compute beta using the S+ Method  and SI Method
Estimate1 <- Estimate.Beta(extended.params = append(param.dip,
                                                    list(C = D1$C)))
SI.Beta <- Estimate1[,6] # SI estimate
S.plus.Beta <- Estimate1[,1] # S+ estimate
SI.S <- Estimate1[,2] # S_t estimate
SI.I <- Estimate1[,3] # I_t estimate

## Now let's plot beta(t) and I(t) and their estimated values
par(mfrow = c(2,1), mar = c(0, 5, 4, 2) + 0.1)
with(param.dip,{
  ## a. plot beta(t) and its estimates
  col1 <- c(brewer.pal(8, 'Accent')[c(8)])
  yrs <- times/52 # time is in weeks, lets plot in years
  ## we want to plot beta in units of R0
  ## so need to multiply it by mult:
  mult <- pop.size/(gamma.val + mu)
  end <- which(yrs > 5.2)[1] # just look at the first 5.2 years
  ## plot the real beta
  plot(yrs[1:end], meas.beta[1:end]*mult, lwd = 5,
       ylim = mult*c(0.95*min(meas.beta), 1.1*max(meas.beta)),
       xlab = "", col = col1, las =1, xaxt = "n",
       ylab = "$\\beta(t)$ (in units of $\\R$)", type = "l",
       main = "Estimating $\\beta(t)$
       (measles parameters with $\\nu(t)$ doubled)")
  ## plot SI method
  lines(yrs[1:end], SI.Beta[1:end]*mult, col = "blue", lwd = 3)
  ## plot S+ method
  lines(yrs[1:end], S.plus.Beta[1:end]*mult, col = "red", lty =3, lwd = 3)
```

```
  grid(col = gray(0.9)) # add a grid
  ## Create a legend
  legend("bottomright",
         c("$\\beta(t)$", "$S^+$ Method Estimate", "$SI$ Method Estimate"),
         col = c(col1, "Red", "Blue"),
         lty = c(1, 3, 1) , lwd = c(5, 3, 3), cex = 0.9, bg = "white")

  ## b. plot I(t) and its estimate I_t
  par(mar = c(4, 5, 0, 2) + 0.1)
  col2 <- "#1b9e77" # turquoise
  col1 <- "#e7298a" # pink
  plot(yrs[1:end], D1$I[1:end], lwd = 3, type = "l", # I(t)
       col = col1, ylab = "",
       xlab = "Time (Years)", las = 1)
  lines(yrs[1:end], SI.I[1:end], col = col2, lwd = 3, lty = 1) #I_t
  mtext("$I(t)$", las = 1 , side = 2, line = 3)
  grid(col = gray(0.9)) # add a grid
  ## Create a legend
  legend("bottomright", c("True $I(t)$", "Estimated $I_t$"),
         col = c(col1, col2),
         lty = c(1, 1), lwd = c(3, 3), cex = 0.9, bg = "white")
})
```



S6.2 `R Code`: Dependence on $\mathcal{R}_0$ and α

The seasonal amplitude α and mean of the transmission rate $\beta(t)$ also affects the performance of each of the methods. Figures 3–4 of the main text, display the error in estimating $\beta(t)$ with measles parameters for a range of $\mathcal{R}_0 \in [0, 30]$ and $\alpha \in [0, 0.1]$. Here we will present the underlying `R` functions used to produce Figures 3–4.

The main function `beta.2D()` calls `comp.error.2d()` to compute the error in estimating $\beta(t)$ for a range of $\mathcal{R}_0$ and α values, and then plots these results using `ColorPlot.2D()`.

```
beta.2D <- function(R0.range, # Range of R0 values
                    amplitude.range, # Range of Alpha values
                    param, # parameter set
                    period, # period of beta(t) (usually 1 yr)
                    noise.percent, # epsilon value
                    five.year.delay = FALSE,
                    max.val.legend = 2, # maximum error plotted
```

```
                    matrix.file.name, # location where the
                                      # estimation error is stored
                    binom.dist = FALSE,
                    outer.title = ""){ # specify overall title
  ## (binom.dist = TRUE) means that we want to simulate case notification
  ## data with a binomial distribution to mimick observation error.

  with(as.list(param), {
    ## (1) Compute all of the error:

    ## If the error has not been previously computed and stored,
    ## use comp.error.2d to compute the error for the range of R0
    ## and alpha.
    cond1 <- file.exists(paste(matrix.file.name, "-SI.csv", sep = ""))
    cond2 <- file.exists(paste(matrix.file.name, "-S+.csv", sep = ""))
    if(cond1 == FALSE | cond2 == FALSE){
    error.output <- comp.error.2d(R0.range = R0.range,
                                  amplitude.range = amplitude.range,
                                  param = param,
                                  period = period,
                                  noise.percent = noise.percent,
                                  five.year.delay = five.year.delay,
                                  binom.dist = binom.dist,
                                  matrix.file.name)
    }

    else { # If the error has already been computed and saved, then
      name1 <- paste(matrix.file.name, "-SI.csv", sep = "")
      name2 <- paste(matrix.file.name, "-S+.csv", sep = "")
      error.output <- list(as.matrix(read.csv(name2, row.names = 1)),
                           as.matrix(read.csv(name1, row.names = 1)))
    }

    ## (2) Plot the error in a 2D plot with a colour gradient representing
    ## the estimation error
    ColorPlot.2D(error.matrices = error.output, x.values = amplitude.range,
                 y.values = R0.range, max.value = max.val.legend,
                 noise.percent = noise.percent, outer.title = outer.title)
  })
}
```

`comp.error.2d` takes in a range of $\mathcal{R}_0$ and α, and for every $\mathcal{R}_0$-α pair, creates a $\beta(t)$ and uses this to simulate case notification data. It then estimates $\beta(t)$ using the S^+ and SI method and the simulated data, and computes the error in estimation. The error is stored in a matrix with rows corresponding to α and columns corresponding to $\mathcal{R}_0$.

```
comp.error.2d <- function(R0.range,
                          amplitude.range,
                          param, period, noise.percent,
                          five.year.delay, binom.dist = binom.dist,
                          matrix.file.name){
  with(as.list(param), {
    n <- length(amplitude.range)
    m <- length(R0.range)

    ## create empty matrices to store the estimation error for the SI
    ## and S plus beta-estimation methods.
    error.SI.start <- matrix(data = rep(0, (n*m)), nrow = n, ncol= m)
    error.S.plus <- matrix(data = rep(0, n*m), nrow = n, ncol = m)

    ## now for every value in amplitude.range and R0.range
    ## we compute the error in estimating Beta
    for (R0.index in 1:m){ # for every R0 value
      for (amp.index in 1:n){ # and every amplitude value

        ## create a parameter list to use
        pass.params <- list(gamma.val = gamma.val, mu = mu,
                            pop.size = pop.size,
                            R0 = R0.range[R0.index],
                            times = times)
        # Create beta(t)
        beta.vec <- Create.Beta(param.list = pass.params,
                                amp = amplitude.range[amp.index],
                                period = period,
                                noise.percent = noise.percent)
        ## lets replace any possible negative beta entry with zero.
        neg.vals <- which(beta.vec < 0)
        beta.vec[neg.vals] <- 0

        ## compute initial conditions for the given R0.
        Init.S <- 1/R0.range[R0.index]
        mean.beta <- R0.range[R0.index]*(gamma.val + mu)/pop.size
        Init.I <- (R0.range[R0.index] - 1)*mu/(mean.beta*pop.size)
        Init.R <- 1 - Init.S - Init.I

        ## place these in a parameter list
        param.use <- replace.param(param, param.index = 10,
                                   new.val = Init.S)
        param.use <- replace.param(param.use, param.index = 11,
                                   new.val = Init.I)
```

```
      param.use <- replace.param(param.use, param.index = 12,
                                 new.val = Init.R)

      #now we compute the SIR model
      SIR.set <- solve.SIR(params = param.use, beta = beta.vec,
                           binom.dist = binom.dist)

      ## Estimate beta(t) using the S+ and SI method
      Est <- Estimate.Beta(append(param.use, list(C = SIR.set$C)))

      ## Concern: In the case of case notification data from the
      ## stochastic SEIR model (when looking at sensitivity
      ## to process error) fadeout is possible.
      ## If there is fadeout, we want the estimate of Beta(t) to
      ## be NA and not 0.  This already happens in the S+ method
      ## (since we get divide by zero), but needs to be adjusted for
      ## the SI method (stored in Estimates[,6])
      no.zeros <- which(Est[,6] == 0)
      Est[,6][no.zeros] <- rep(NA, length(no.zeros))

      ## If more than 20\% of the values in an estimate are NA, we change
      ## the error term to be 0. Then, when plotting the R0-alpha values
      ## that cause a 0 error term is marked in gray.  We do not want to
      ## be mislead that estimates have small error if
      ## they are NA most of the time.
      na.percents <- count.nas(Est) # count the number of NA values
      temp <- error.est(est.data = Est, real.data = SIR.set,
                        five.year.delay = five.year.delay)
      for (index in 1:length(na.percents)){
        if (na.percents[index] >= 20){
          temp[index] <- 0
        }
      }

      ## Then store the estimation methods in the matrices
      error.SI.start[amp.index, R0.index] <- temp[6]
      error.S.plus[amp.index, R0.index] <- temp[1]

      ## Check if I(t) < 0 at some point, if so this will also
      ## be plotted in white, so we know something is going on.
      ## It also prints warning messages.
      I.check <- which(SIR.set$I < 0 )
      if (length(I.check) > 0){
        print(paste("negative I values at mean value: ",
```

```
                        R0.range[R0.index], "at R0.index: ",
                        R0.index, "amplitude: ",
                        amplitude.range[amp.index]))
          print(paste("min value: ", min(SIR.set$I, na.rm = TRUE)))
          error.SI.start[amp.index, R0.index] <- -1e-16
          error.S.plus[amp.index, R0.index] <- -1e-16
        }
      } # end of ranging through the amplitudes
    } # end of ranging through the R0 values.

    ## save the matrices holding the error information to .csv filed
    name1 <- paste(matrix.file.name, "-SI.csv", sep = "")
    name2 <- paste(matrix.file.name, "-S+.csv", sep = "")
    write.csv(error.SI.start, name1)
    write.csv(error.S.plus, name2)

    ## return the computed error matrices
    return(list(error.S.plus, error.SI.start))
  })
}
```

`ColorPlot.2D` takes in the matrices computed by `comp.error.2d` and plots them, with α on the x-axis and $\mathcal{R}_0$ on the y-axis. For each $\mathcal{R}_0$-α pair, a point is plotted with a colour corresponding to the RRMSE. `calc.col()` is a sub-function that chooses the colour to plot a point with given the RRMSE in each case.

```
ColorPlot.2D <- function(error.matrices, # the 2 error matrices
                         x.values, # x values in our plot (amplitude)
                         y.values,  # y values in our plot (mean value)
                         max.value,  # max error to be plotted
                         noise.percent = 0, # percent noise in beta
                         outer.title = ""){ # outer title for the plot
  par(oma = c(1, 2, 2, 2))
  ## set up the plotting area.
  layout(matrix(c(1, 1, 1, 1, 2, 2, 2, 2, 3), nrow = 1, ncol = 9,
                byrow = TRUE))

  ## find the maximum value for the error legend.
  max.vals <- rep(NA, 2)
  min.vals <- rep(NA, 2)
  index <- 1
  for (matrix in error.matrices){
    max.vals[index] <- max(matrix[which(matrix < max.value)])
    min.vals[index] <- min(matrix)
```

```
  index <- index + 1
}

max.val <- max(max.vals)
min.val <- min(min.vals)

titles <- c("$S^+$ Method", "$SI$ Method") # plot titles

## plot the results
index <- 1
par(mar = c(5, 4, 4, 0) + 0.1)
for (matrix in error.matrices){
  ## Set up the plotting window
  if (index == 1){ # If index = 1, keep the y-axis
    plot(0, 0, type = "l", col = "white",
         ylim = c(min(y.values), max(y.values)),
         xlim = c(min(x.values), max(x.values)),
         xlab = "$\\alpha$", ylab = "",
         main = titles[index], las = 1, cex.lab = 1.5)
    mtext("$\\R$", side = 2, line = 3, las = 1)}
  if (index == 2){ # if index = 2, don't print the y-axis
    par(mar = c(5, 1, 4, 3) + 0.1)
    plot(0, 0, type = "l", col = "white",
         ylim = c(min(y.values), max(y.values)),
         xlim = c(min(x.values), max(x.values)),
         xlab = "$\\alpha$",
         main = titles[index], yaxt = "n", las =1, cex.lab = 1.5)
  }
  ## then for each entry in the matrix plot a point
  ## representing the estimation error
  for (col.index in 1:(dim(matrix)[2])){ # for every matrix column (RO)
    for (row.index in 1:(dim(matrix)[1])){# for every matrix row (alpha)
      ## first determine the color using the function calc.col
      val <- matrix[row.index, col.index]
      p <- calc.col(val, max.val, min.val)
      ## then plot the point
      points(x.values[row.index], y.values[col.index],
             col = p, pch = 15, cex = 2.1)
    }
  }
  index <- index + 1
}
## Then produce an overall title if outer.title is not ""
if (!outer.title == ""){
```

```
    title(outer.title, outer = TRUE, cex.main = 2)
  }
  ## create a color legend on the side.

  list.of.colors <- blue2green2red(51)
  list.of.vals <- seq(min.val, max.val, length.out = 51)
  par(mar = c(5, 1, 4, 4) + 0.1)
  plot(0, 0, col = "white", ylim = c(min.val, max.val),
       xlab = "", main = "RRMSE", xaxt = "n",
       ylab = "", yaxt = "n", las = 1, cex.main = 0.85)
  axis(4, las = 1)
  for (index in 1:length(list.of.vals)){
    points(0, list.of.vals[index], col = list.of.colors[index],
           pch = 15, cex = 2)
  }

}
```

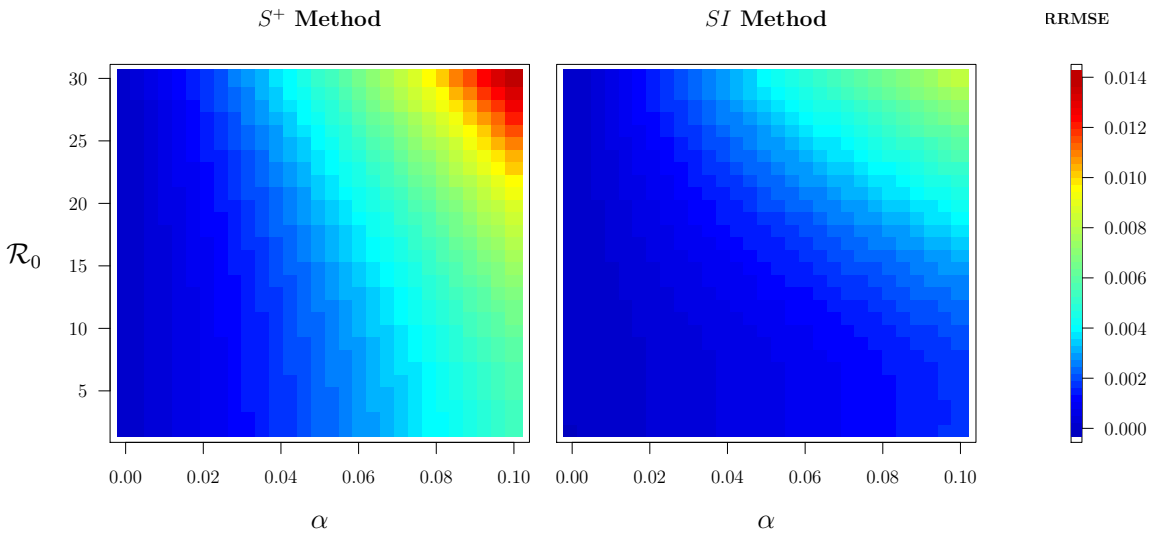
```
calc.col <- function(val, # value to assign a color to
                     max.val, # min and max error values
                     min.val){

  library("colorRamps")
  ## if val = 0, the estimate has more than 20\% NA values
  if (val == 0){
    plot.col <- "gray"
  }
  ## if val = -1e-16 the I(t) term has negative parts
  else if (val == -1e-16){
    plot.col <- "white"
  }
  ## if value is greater than the max value to put on the legend
  else if (val  >  max.val){
    plot.col <- "black"
  }
  else{  # otherwise: use the blue2green2red color scheme
    list.of.colors <- blue2green2red(51)
    list.of.vals <- seq(min.val, max.val,
                        length.out =  length(list.of.colors))
    k <- which(list.of.vals >= val)[1]
    plot.col <- list.of.colors[k]
  }
  return(plot.col)
```

```
}
```

S6.2.1 Dependence on $\mathcal{R}_0$ and α with smallpox parameters

Figures 3–4 of the main text display how the error in estimation depends on the values of $\mathcal{R}_0$ and α for measles parameters. We will reproduce the same figures here, but for smallpox parameters, looking both at a smooth and noisy $\beta(t)$ ($\epsilon = 0$ and $\epsilon = 0.5$ respectively).

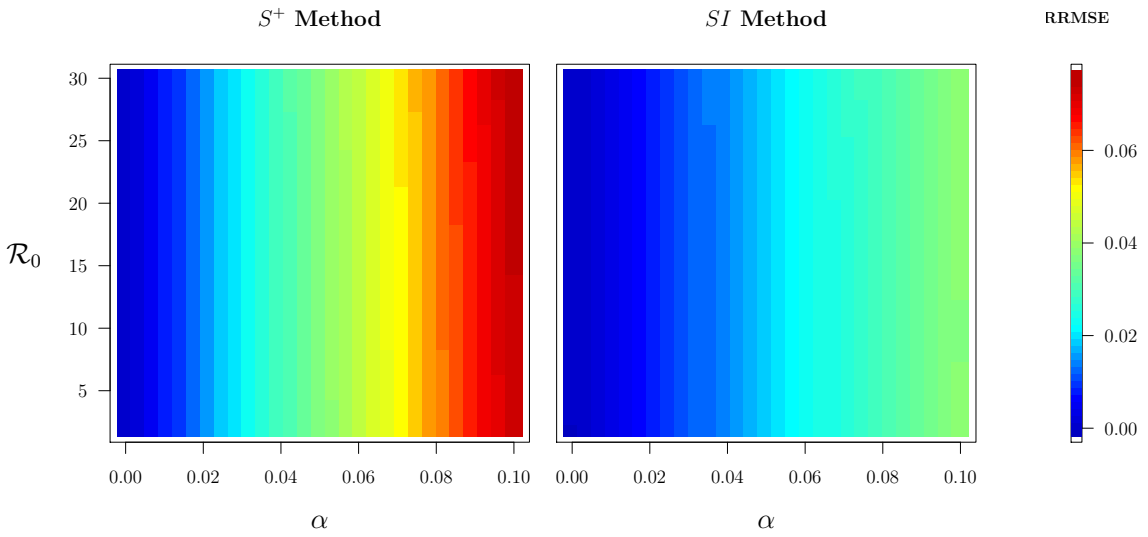
```
## 1. Smooth transmission rate beta(t)
RR <- seq(2, 30, by = 1) # range of R0 values
amp <- seq(0, 0.1, length.out = length(RR)) # range of alpha values

## compute and plot the estimation error
beta.2D(R0.range = RR, amplitude = amp,
        param = param.small, # smallpox parameters
        period = 1, noise.percent = 0,
        five.year.delay = FALSE, matrix.file.name = "Smallpox-0noise")
```



This figure produced with smallpox parameters looks very similar to Figure 3 which was produced for measles parameters, except that the overall error estimation scale here is smaller. As in Figure 3 the SI method performs better overall, and large values of $\mathcal{R}_0$ and α decrease estimation accuracy for both methods.

```
## 2. Noisy transmission rate beta(t)
RR <- seq(2, 30, by = 1)
amp <- seq(0, 0.1, length.out = length(RR))
```

```
## Smallpox, zero noise:
beta.2D(R0.range = RR, amplitude = amp,
        param = param.small, # smallpox parameters
        period = 1, noise.percent = 0.5,
        five.year.delay = FALSE, matrix.file.name = "Smallpox-50noise")
```



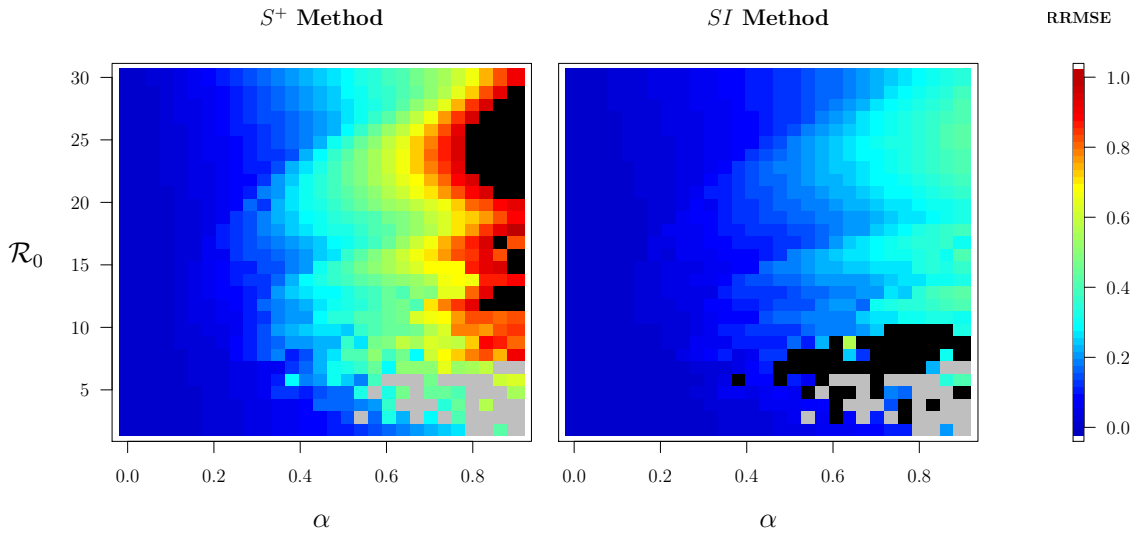
This figure for smallpox parameters looks qualitatively identical to Figure 4 of the main text which was produced with measles parameters. For this noisy transmission rate, the estimation error increases quickly with large α since noise is added to $\beta(t)$ in proportion to α.

S6.2.2 Estimating transmission rates with large amplitude

Up to think point, we have looked at how estimation accuracy depends on the seasonal amplitude α of transmission for $\alpha \in [0, 0.1]$. This is because α is estimated to be 0.08 for measles [1], and between $0.032 - 0.12$ for smallpox (in London, England from 1664 - 1930) [3]. However, we are interested to see how each estimation method performs for transmission rates with large seasonal amplitude. Here we examine the estimation error for $\alpha \in [0, 0.9]$. First, we look at the case with a smooth transmission rate ($\epsilon = 0$ in Equation (S11)).

```
## Case 1: Smooth beta(t)
RR <- seq(2, 30, by = 1) # R0 range
amp.large <- seq(0, 0.9, length.out = length(RR)) # alpha range
beta.2D(R0.range = RR,
        amplitude.range = amp.large,
        param = param.meas, period = 1,
        noise.percent = 0, max.val.legend = 1,
        five.year.delay = FALSE,
```

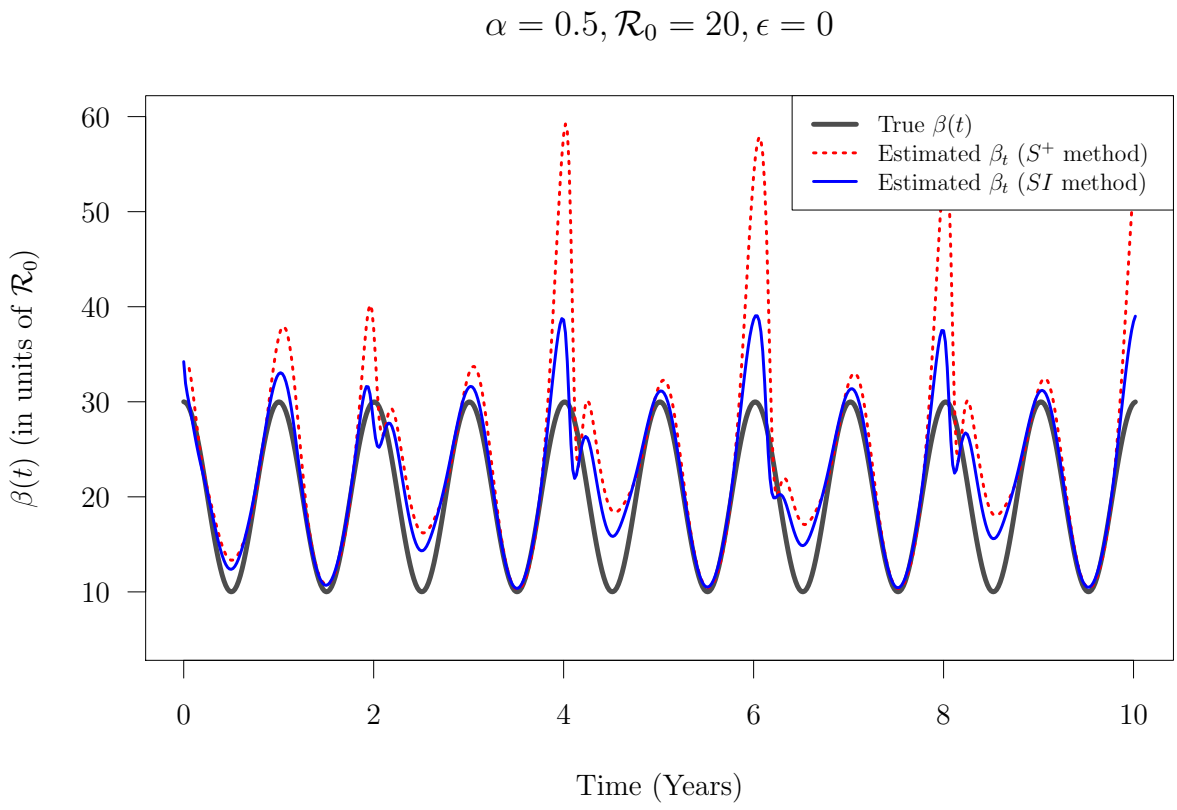
```
        matrix.file.name = "Measles-0noise-largealpha-3")
```



The grey squares represent a $\beta(t)$-estimate that is NA for more than 20% of the time. This usually means that for many time intervals Δt there is no case notifications. The black squares mean that the error in estimation is greater than one. Here we see an interesting wave pattern when looking at estimation error as a function of $\mathcal{R}_0$. There is a clear advantage to using the SI method for estimation, except in the case of small $\mathcal{R}_0$ (< 10) and very large α, where there is a lot of error in the SI method. In all other places the SI method outperforms the S^+ method. In order to gain a clearer understanding of the estimation accuracy displayed in this plot, let's look at the actual estimates for $\beta(t)$ with the S^+ and SI methods and measles parameters if $\mathcal{R}_0 = 20$ and $\alpha = 0.5$.

```
## define parameters and beta
param.meas <- param.define(type = "Measles", no.years = 20)
case1.beta <- Create.Beta(param.list = param.meas,
                          amp = 0.5, period = 1, noise.percent = 0)
## solve the SIR model
Case1 <- solve.SIR(param = param.meas, beta = case1.beta)
## estimate beta using the S+ and SI methods
Estimates1 <- Estimate.Beta(append(param.meas, list(C = Case1$C)))
## now plot Beta(t) and the estimates
par(mfrow = c(1,1))
with(param.meas, {
  end <- which(times > 10*52)[1]
  mult <- pop.size/(gamma.val + mu)
  max1 <- mult*max(case1.beta)*2
  min1 <- mult*min(case1.beta)/2
```

```
  ## plot the true beta(t)
  plot(times[1:end]/52, mult*case1.beta[1:end], lwd = 5, type = "l",
       main = "$\\alpha = 0.5, \\R = 20, \\epsilon = 0$",
       ylim = c(min1, max1), xlab = "Time (Years)",
       ylab = "$\\beta(t)$ (in units of $\\R$)", col = gray(0.3), las = 1)
  ## plot the S+ method estimate
  lines(times[1:end]/52, mult*Estimates1[,1][1:end],
        col = "red", lty = 3, lwd = 3)
  ## plot the SI method estimate
  lines(times[1:end]/52, mult*Estimates1[,6][1:end],
        col = "blue", lwd = 3)
  ## add a legend
  legend("topright",
         c("True $\\beta(t)$", "Estimated $\\beta_t$ ($S^+$ method)",
                       "Estimated $\\beta_t$ ($SI$ method)"),
         col = c(gray(0.3), "red", "blue"), lwd = c(5,3,3),
         lty = c(1,3,1), bg = "white", text.width = 3, cex = 0.8)
})
```

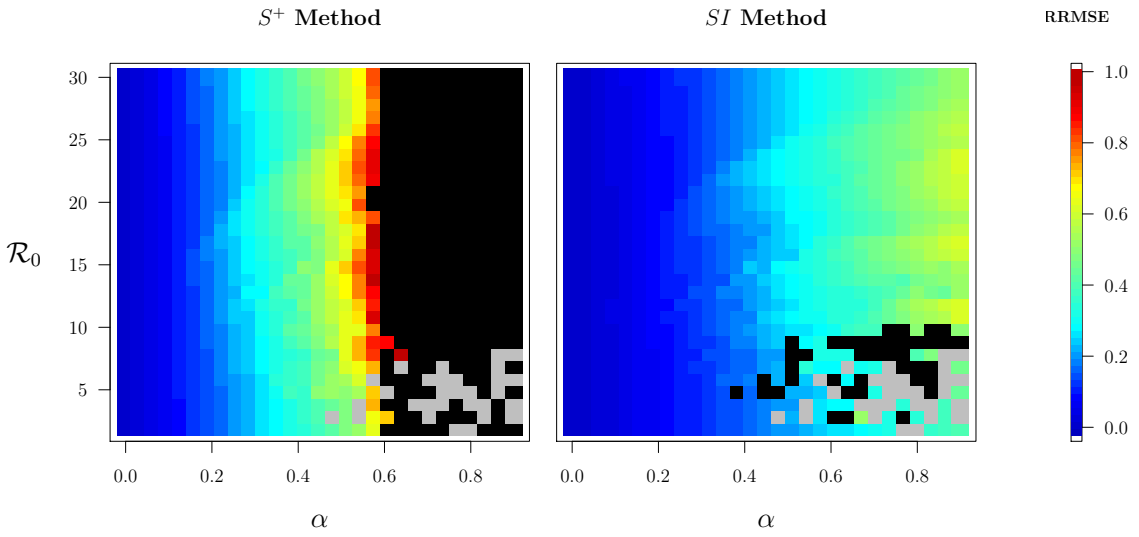


So we see that estimating transmission rates with such large seasonal amplitude produce some interesting results. Here both methods are producing an estimated transmission rate

that has a period longer than one year.

Next, we will look at the dependency of estimation error for $\alpha \in [0, 0.9], \mathcal{R}_0 \in [0, 30]$ if we use a 'noisy' transmission rate ($\epsilon = 0.5$ in Equation (S11)).

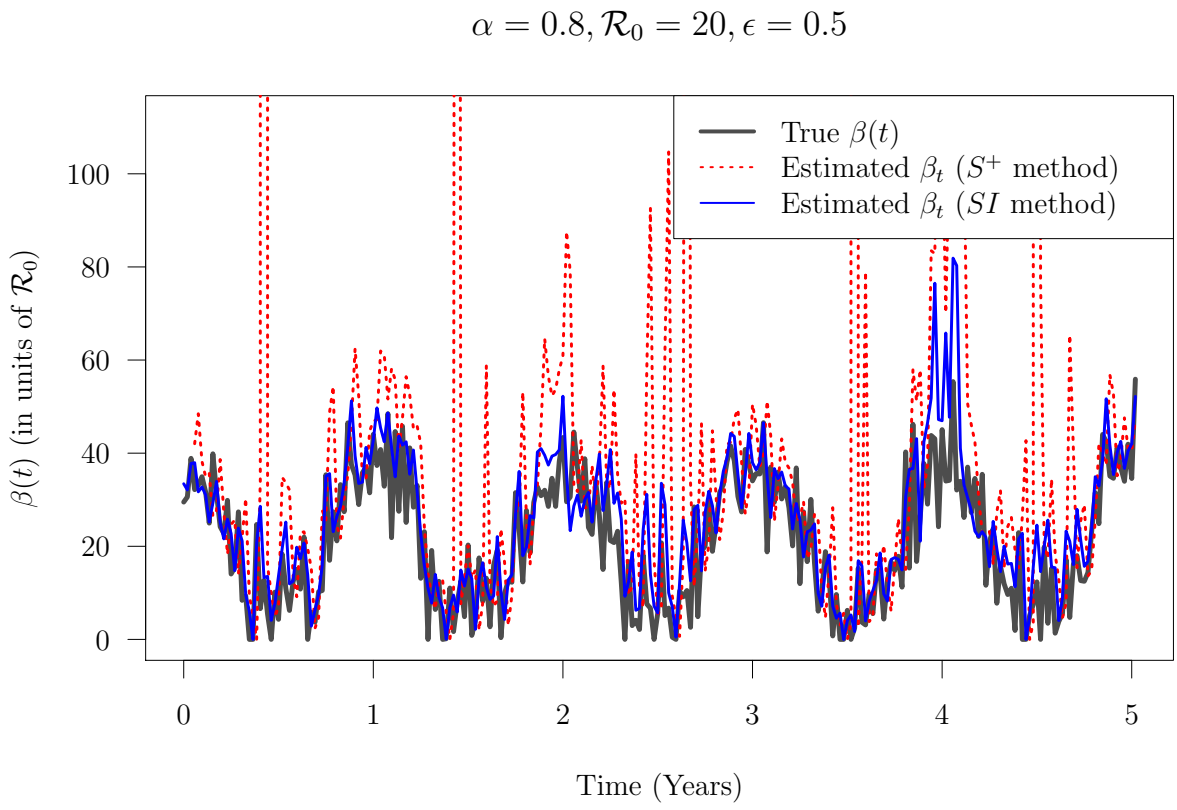
```
beta.2D(R0.range = RR,
        amplitude.range = amp.large,
        param = param.meas, period = 1,
        noise.percent = 0.5, max.val.legend = 1,
        five.year.delay = FALSE,
        matrix.file.name = "Measles-50noise-largealpha-3")
```



Since the noise is added as a proportion of the amplitude α, estimation accuracy decreases quickly for large values of α simply because the underlying $\beta(t)$ is much noisier. Here we see a clear advantage in using the SI method. We will look at one specific set of estimates for the 'noisy' transmission rate with $\mathcal{R}_0 = 20$ and $\alpha = 0.8$.

```
## use measles parameters, choose alpha = 0.8, epsilon = 0.5
param.meas <- param.define(type = "Measles", no.years = 20)
case2.beta <- Create.Beta(param.list = param.meas, amp = 0.8,
                          period = 1, noise.percent = 0.5)
## solve the SIR model in each case
Case2 <- solve.SIR(param = param.meas, beta = case2.beta)
## estimate beta(t)
Estimates2 <- Estimate.Beta(append(param.meas, list(C = Case2$C)))
## now plot Beta(t) and the estimates
par(mfrow = c(1,1))
with(param.meas, {
  end <- which(param.meas$times > 52*5)[1]
```

```
  mult <- pop.size/(gamma.val + mu)
  max2 <- mult*max(case2.beta)*2
  min2 <- mult*min(case2.beta)/2
  plot(times[1:end]/52, mult*case2.beta[1:end], lwd = 5, type = "l",
       main = "$\\alpha = 0.8, \\R = 20, \\epsilon = 0.5$",
       ylim = c(min2, max2), xlab = "Time (Years)",
       ylab = "$\\beta(t)$ (in units of $\\R$)",
       col = gray(0.3), las = 1)
  lines(times[1:end]/52, mult*Estimates2[,1][1:end],
        col = "red", lty = 3, lwd = 3)
  lines(times[1:end]/52, mult*Estimates2[,6][1:end],
        col = "blue", lwd = 3)
  legend("topright",
         c("True $\\beta(t)$", "Estimated $\\beta_t$ ($S^+$ method)",
                       "Estimated $\\beta_t$ ($SI$ method)"),
         col = c(gray(0.3), "red", "blue"), lty = c(1,3,1),
         lwd = c(4,2,2), bg = "white", text.width = 2)
})
```



This plot clearly emphasizes the benefit of using the SI method in the case of a noisy transmission rate. Even with very large seasonal amplitude, the transmission rate is predicted

accurately by the SI method, but the S^+ method predicts some very large values for the transmission rate.

S7 Sensitivity to incorrect parameters

Many parameter values necessary for these fast estimation methods are unknown (*e.g.,* reporting ratio, initial number of susceptibles, etc), and are difficult to estimate. In this section, we explore how well the $\beta(t)$-estimation methods perform if they estimate $\beta(t)$ with incorrect parameter values. Starting from a base set of measles or smallpox parameters, the simulated case notification data is generated. Then, the S^+ and SI methods are given this simulated data and a set of parameters, where one is incorrect. We measure the estimation error for each method with an incorrect parameter, in order to get an idea of which parameter values are essential to know for accurate estimation of the transmission rate.

S7.1 `R Code`: Sensitivity to incorrect parameters

The main function that computes the estimation error for each set of parameter values and plots the results is named `ParamIncorrect()`. It calls many of the functions previously defined in §S6.1 (the structure is very similar to the tree in §S6.1) but `compute.RRMSE.Incorrect()` is used instead of `compute.RRMSE()`

```
ParamIncorrect <- function(beta, orig.param, # transmission rate and params
                           five.year.delay = FALSE,
                           method.name, # either "S+" or "SI.start"
                           min.error.val = 0,
                           max.error.val, # min/max error plotted
                           max.percent,
                           min.percent, # min/max percent range
                          right = FALSE, # if yaxis should go on the right
                          stored.data.name, # name where we store the data
                          param.name, ...){ # name of the parameter set
  ## If the estimation error has been computed for all the parameters sets
  ## then we just load the saved data in stored.data.name
  ## But if that file doesn't exist, we will go through the process of
  ## computing estimation error.
  if (!file.exists(stored.data.name)){
    ## The error as a function of the incorrect parameter value is stored in
    ## Error.Param.List.  Each entry corresponds to a data frame for the
    ## parameter indicated by param.index that contains the parameter range
    ## of incorrect values and the error in estimation
    Error.Param.List <- vector(mode = "list", length = 7)
    EPM.i <- 1  # index of the Error.Param.List
    ## number of points we want in each range:
    length.out <- 15
```

```
    ## then for each parameter, vary one parameter from 50% to 200% of
    ## its correct value, and compute the error in using that incorrect
    ## parameter
    for (param.index in c(3, 4, 5, 7, 9, 10, 11)){
      # first get the range of parameter values and percentange range
      output <- Parameter.Range(max.percent = max.percent,
                                min.percent = min.percent,
                                orig.param = orig.param,
                                param.index = param.index,
                                length.out = length.out,
                                incorrect = TRUE)
      ## compute the error for each value in the parameter range
      error.vec <- compute.RRMSE.Incorrect(orig.param = orig.param,
                                           param.index = param.index,
                                           range = output[[1]],
                                           method.name = method.name,
                                           five.year.delay = five.year.delay,
                                           beta = beta, mu.only = TRUE)
      ## store that error in Error.Param.List
      Error.Param.List[[EPM.i]] <- data.frame(percent.range = output[2],
                                              error = error.vec)
      ## increase the index of Error.Param.List by 1
      EPM.i <- EPM.i + 1
    }
    ## save this data set for next time
    save(Error.Param.List, file = stored.data.name)
  }

  ## define a list of colours
  col.list <- c("slateblue2", "darkslategray3", "darkseagreen4",
                "gray20", "deeppink3", "mediumorchid2", "tan1")
  ## state the legend label names
  label.names <- c("$ \\gamma^{-1}$" , "$ \\mu $", "$\\Trep$",
                   "$\\rho \\eta$",  "$ \\nu $","$ S_0 $", "$ I_0 $")
  ## Plot the data using Plot.Params
  Plot.Params(stored.data.name = stored.data.name,
                  col.list = col.list,
                  label.names = label.names,
                  min.error.val = min.error.val,
                  max.error.val = max.error.val,
                  min.percent = min.percent, max.percent = max.percent,
                  x.spot = 1.7, y.spot = 0.95, right = right,
                  param.name = param.name,  ...)
}
```

`compute.RRMSE.Incorrect()` takes in a transmission rate $\beta(t)$ and a set of 'true' parameters (measles or smallpox) and generates simulated case notification data. Then, using a parameter set that contains one incorrect parameter at a time, it computes the RRMSE in estimation of $\beta(t)$.

```
compute.RRMSE.Incorrect <- function(orig.param, # parameter set
                                    param.index,  # parameter to explore
                                    range, # parameter range
                                    method.name, # S+ or SI.start
                                    five.year.delay,
                                    beta, # beta (t)
                                    mu.only){ # vary mu independantly of nu
  ## Compute the SIR and case notification data with the true parameters
  SIR <- solve.SIR(params = orig.param, beta = beta)
  ## for each element in the parameter range, we create a parameter set
  ## that contains this element as the parameter value for param.index.
  ## Then each of these parameter sets are stored together in a list:
  ## param.range.list
  param.range.list <- range.list(orig.param = orig.param,
                                 param.index = param.index, range = range)
  ## since a change in one parameter sometimes implies a change in another
  ## parameter PRL.implications goes through and implements all these changes
  param.range.list <- PRL.implications(orig.param = orig.param,
                                       param.range.list = param.range.list,
                                       param.index = param.index,
                                       mu.only = mu.only,
                                       range = range,
                                       ## incorrect = TRUE if we are
                                       ## dealing with incorrect parameters
                                       incorrect = TRUE)
  ## create an vector to store the estimation error
  error.vec <- rep(NA, length(param.range.list))
  index <- 1 # index of the error vector
  ## Then estimate beta using the parameters in param.range.list
  ## and compute the estimation error
  for (param.set in param.range.list){ # for each parameter set
    ## combine the parameter set and case notification data:
    e.p <- append(param.set, list(C = SIR$C))
    ## estimate Beta, S, I using the S+ and SI method
    Estimates <- Estimate.Beta(e.p)
    ## count the number of NA entries in the Estimates
    na.percents <- count.nas(Estimates[1])[1]
    ## compute the error in estimation
    error <- error.est(est.data = Estimates, # estimated beta/S/I
                       real.data = SIR,  # true beta/S/I
```

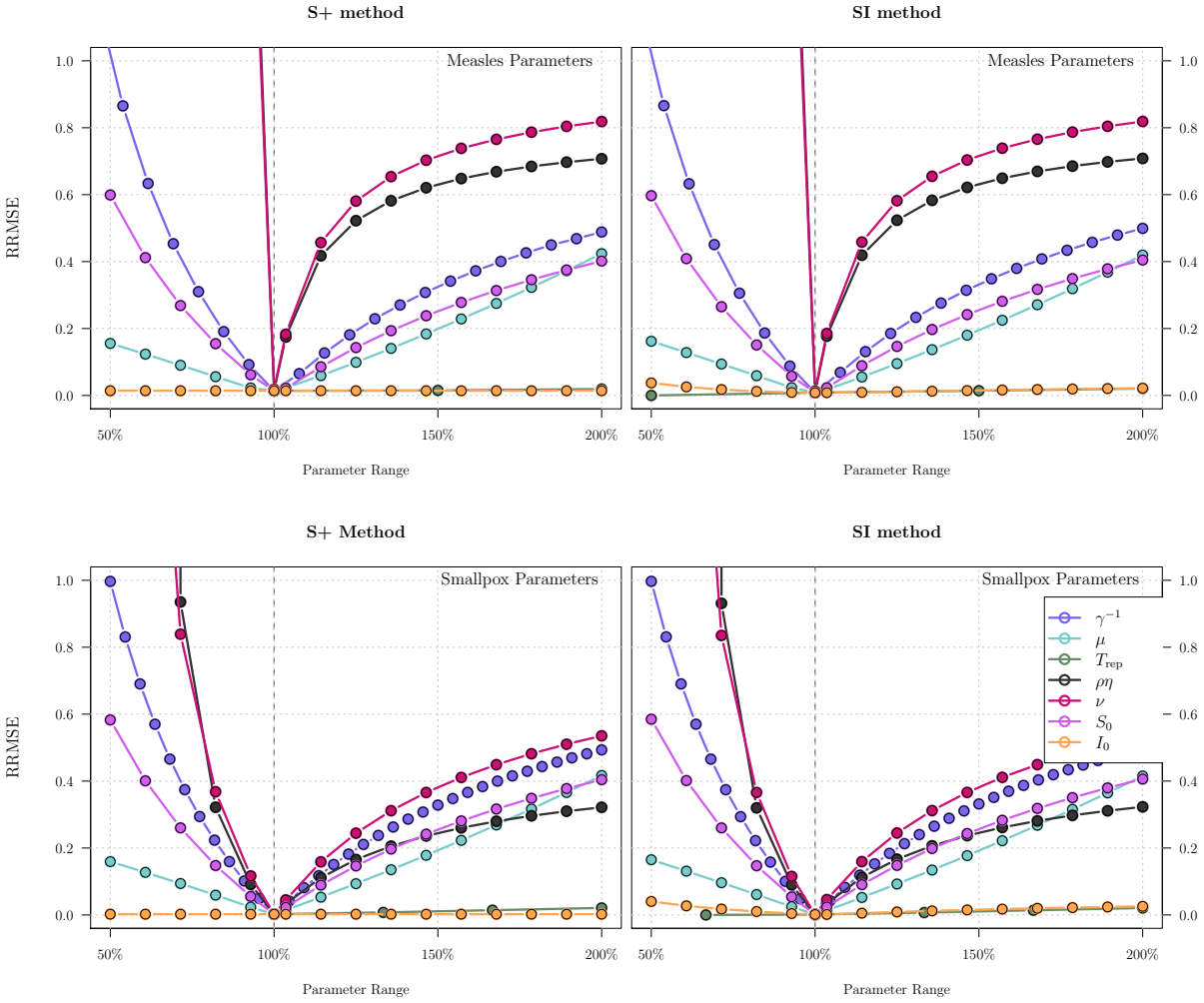
```
                        five.year.delay = five.year.delay)
    ## then if more than 20% of the estimate is NA values,
    ## we replace the error term with an NA value
    for (index.na in 1:length(na.percents)){
      if (na.percents[index.na] >= 20){
        error[index.na] <- NA
      }
    }
    ## add the estimation error to the vector
    if (method.name == "SI.start"){
      error.vec[index] <- error[6]
      index <- index + 1
    }
    else if (method.name == "S+"){
      error.vec[index] <- error[1]
      index <- index + 1
    }
  }
  return(error.vec)
}
```

S7.2 Comparing sensitivity to incorrect parameter values between the S^+ and SI methods

In the main text, Figure 5 shows the sensitivity of the SI estimation method to incorrect parameters for smallpox and measles. The S^+ method is not presented in Figure 5 since the difference in error between the S^+ and SI method is much smaller than the error obtained by using incorrect parameter values. Here, we show sensitivity to incorrect parameter values for measles and smallpox parameters for both the S^+ and SI methods. From the produced figure it is clear that sensitivity of both methods to incorrect parameters is very similar.

```
par(mfrow = c(2,2))
## Case 1: Measles Parameters, S+ Method
par(mar = c(5, 5, 4, 0.2) + 0.1)
ParamIncorrect(beta = meas.beta,  orig.param = param.meas,
               method.name = "S+", min.error.val = 0,
               max.error.val = 1, max.percent = 2, min.percent = 0.5,
               stored.data.name = "MeasIncorrect-Splus.Rdata",
               param.name = "Measles Parameters", main = "S+ method")

## Case 2: Measles Parameters, SI Method
par(mar = c(5, 0.2, 4, 5) + 0.1)
ParamIncorrect(beta = meas.beta, orig.param = param.meas,
```

```
                method.name = "SI.start", min.error.val = 0,
                max.error.val = 1, max.percent = 2, min.percent = 0.5,
                stored.data.name = "MeasIncorrect2.Rdata",
                param.name = "Measles Parameters", main = "SI method",
                right = TRUE, yaxt = "n") # draw y-axis on right side of plot

## Case 3: Smallpox Parameters, S+ Method
par(mar = c(5, 5, 4, 0.2) + 0.1)
ParamIncorrect(beta = small.beta, orig.param = param.small,
                method.name = "S+", min.error.val = 0,
                max.error.val = 1, max.percent = 2, min.percent = 0.5,
                stored.data.name = "SmallIncorrect-Splus.Rdata",
                param.name = "Smallpox Parameters",  main = "S+ Method")

## Case 4: SmallPox Parameters, SI Method
par(mar = c(5, 0.2, 4, 5) + 0.1)
ParamIncorrect(beta = small.beta,  orig.param = param.small,
                method.name = "SI.start", min.error.val = 0,
                max.error.val = 1, max.percent = 2, min.percent = 0.5,
                stored.data.name = "SmallIncorrect2.Rdata",
                param.name = "Smallpox Parameters", main = "SI method",
                right = TRUE, yaxt ="n", legend = TRUE) # add a legend
```

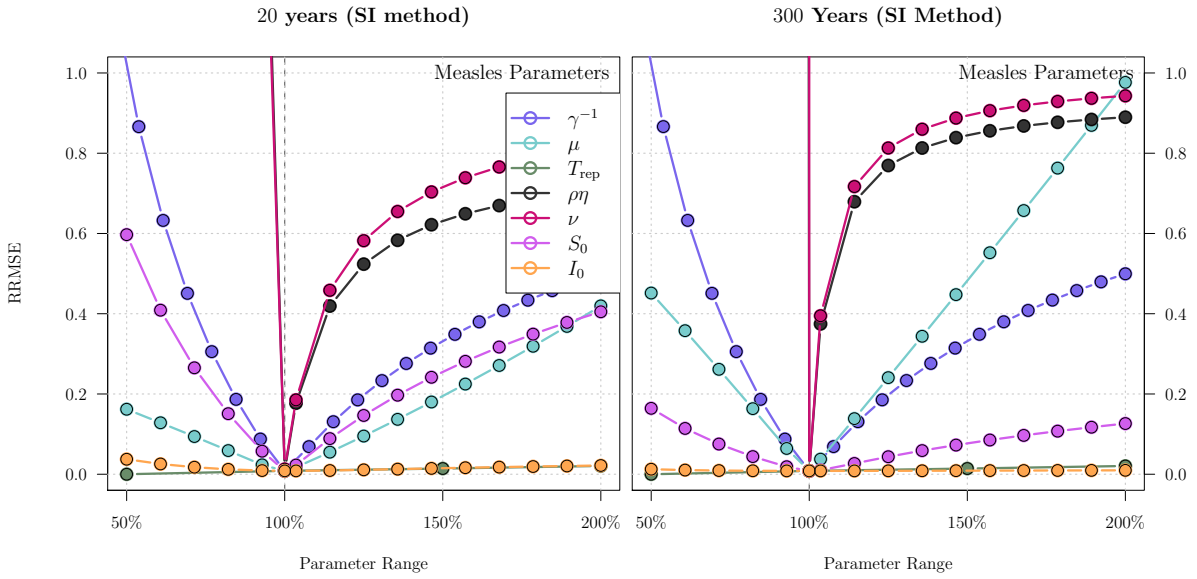


S7.3 Sensitivity of incorrect parameters when estimating the transmission rate for 20 or 300 years

The above plot demonstrates the estimation error that results when estimating the transmission rate with an incorrect parameter over 20 years of weekly simulated case notification data. What would happen if we instead estimated the transmission rate with an incorrect parameter for over 300 years of weekly data? Which incorrect parameters would cause error in estimation to increase over this much longer data set? Here we compare sensitivity to incorrect parameters when the transmission rate is estimated for 20 or 300 years.

```
## First lets plot the 20 year error plot:
```

```
par(mar = c(5, 5, 4, 0.2) + 0.1, mfrow = c(1,2))
ParamIncorrect(beta = meas.beta, orig.param = param.meas,
               method.name = "SI.start", min.error.val = 0,
               max.error.val = 1, max.percent = 2, min.percent = 0.5,
               stored.data.name = "MeasIncorrect2.Rdata",
               param.name = "Measles Parameters",
               main = "$20$ years (SI method)",
               legend = TRUE)

## Then look at error over 300 years:
## define a 300 year set of measles parameters
param.long <- param.define(type = "Measles", no.years = 300)
with(param.long,{
  ## Create a sinusoidal beta(t) for 300 years
    ## compute the mean value of beta
    b0 <- R0*(gamma.val + mu)/pop.size # mean value of Beta
    ppy <- (365/7) # data points per year
    ## compute beta.vec
    beta.vec <- b0*(1 + 0.08*cos(2*pi*times/(ppy)))

## Plot the estimation error
par(mar = c(5, 0.2, 4, 5) + 0.1)
ParamIncorrect(beta = beta.vec,  orig.param = param.long,
               method.name = "SI.start", min.error.val = 0,
               max.error.val = 1, max.percent = 2, min.percent = 0.5,
               stored.data.name = "MeasIncorrect-SI-Long.Rdata",
               param.name = "Measles Parameters",
               main = "$300$ Years (SI Method)",
               right = TRUE, yaxt = "n")
})
```



So here we see that after 300 years, the estimation error incurred by incorrect estimates of $\rho\eta$, ν, and μ has compounded. We particularly see that over this length of time estimation accuracy becomes much more sensitive to an incorrect estimate for the natural mortality rate μ. Also, sensitivity to S_0 decreases over this length of time, since eventually the susceptible update equation will depend very little on S_0.

S8 Sensitivity to observation error

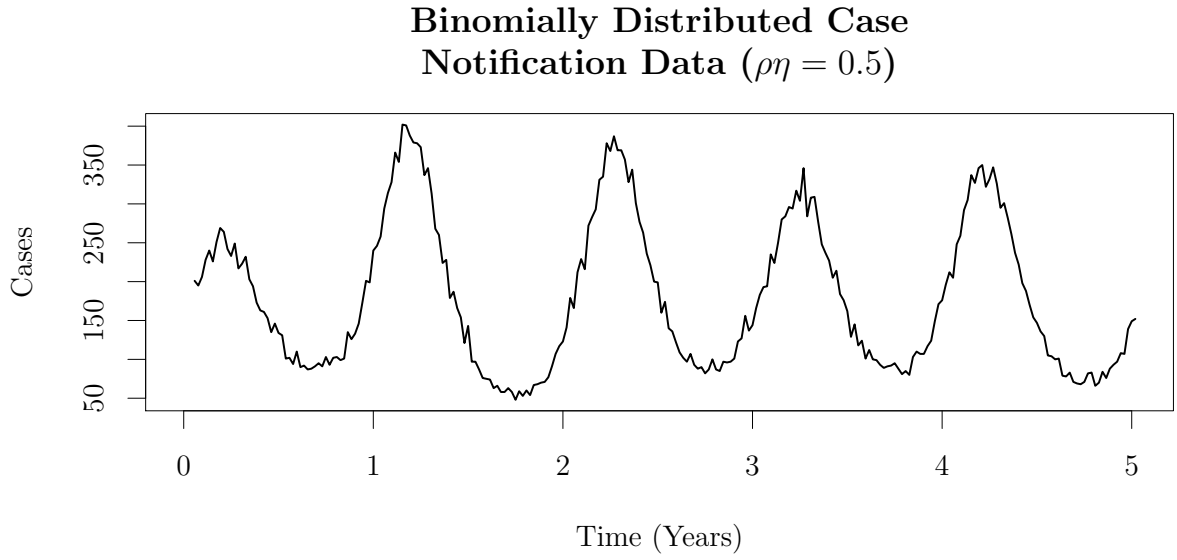
Up until this point, the case notification data has been simulated using the SIR model (Equation (1)), which produces unrealistically smooth and clean data. In order to mimic the observation error that is present in data sets, we assume that the case notification data is sampled from a binomial distribution, where the probability of a case being recorded is equal to the reporting/case fatality ratio ($\rho\eta$). In order to implement this in the already defined `R` functions, we just need to set the marker `binom.dist = TRUE` in the `solve.SIR()` function when generating the simulated case notification data.

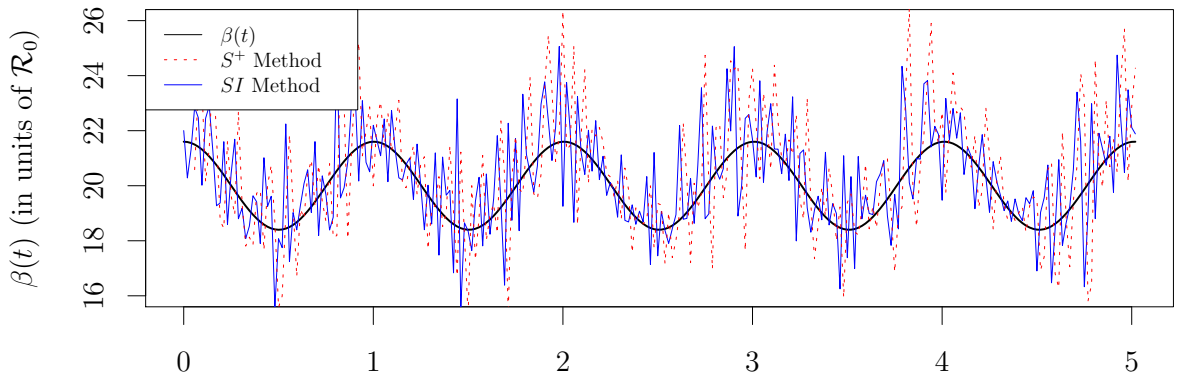
Figure 6 of the main text displays how the S^+ and SI methods perform in the presence of observation error for measles parameters, a reporting/case fatality ratio of 0.2 and a range of $\alpha \in [0, 0.1], \mathcal{R}_0 \in [0, 30]$.

S8.1 An example of estimating $\beta(t)$ with observation error

To visualize the estimation error it is helpful to look at a sample estimate of $\beta(t)$ in the presence of observation error. Here we choose the case with measles parameters and the reporting/case fatality ratio $\rho\eta = 0.5$.

```
## measles parameters
cf.params <- param.meas
cf.params$cf.RR <- 0.5 # reporting/case fatality ratio of 0.5
## Compute the SIR data set, and the simulated case notification data
## Select binom.dist = TRUE
CF.data <- solve.SIR(params = cf.params, beta = meas.beta,
                     binom.dist = TRUE)
## Compute beta using the S+ Method  and SI Method
Estimate <- Estimate.Beta(extended.params = append(cf.params,
                                                   list(C = CF.data$C)))
SI.Beta <- Estimate[,6] # SI estimate
S.plus.Beta <- Estimate[,1] # S+ estimate
## Plot beta(t) and the estimates
par(mfrow = c(2,1), oma = c(0,0,0,0)) # two panel plot
with(cf.params,{
  ## 1. Plot the simulated case notification data:
  yrs <- times/52 # time is in weeks, lets plot in years
  end <- which(yrs > 5)[1]
  bin.title <- "Binomially Distributed Case
                Notification Data ($\\rho \\eta = 0.5$)"
  plot(yrs[1:end], CF.data$C[1:end], type = "l", lwd = 2, ylab = "Cases",
       main = bin.title, xlab = "Time (Years)")
  ## 2. Plot beta(t) and the estimates.
  ## Plot beta in units of R0, so need to multiply beta by mult:
  mult <- pop.size/(gamma.val + mu)
  ## plot the real beta
  plot(yrs[1:end], meas.beta[1:end]*mult, lwd = 2, ylim = c(16, 26),
       xlab = "", ylab = "$\\beta(t)$ (in units of $\\R$)", type = "l",
       main = "Estimating $\\beta(t)$ with Observation Error")
  ## plot SI method
  lines(yrs[1:end], SI.Beta[1:end]*mult, col = "blue", lwd = 0.8)
  ## plot S+ method
  lines(yrs[1:end], S.plus.Beta[1:end]*mult, col = "red", lty =3, lwd = 1)
  ## Create a legend
  legend("topleft", c("$\\beta(t)$", "$S^+$ Method", "$SI$ Method"),
         col = c("Black", "Red", "Blue"), lty = c(1, 3, 1),
         lwd = c(1, 0.4, 0.6), cex = 0.7, bg = "white")
  })
```



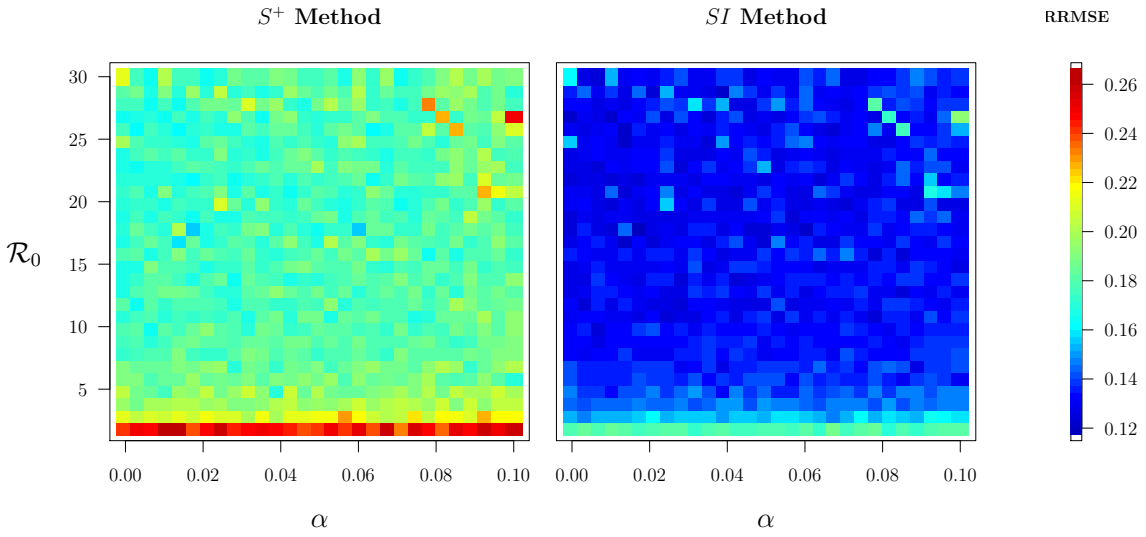


For $\eta\rho = 0.5$, estimation of $\beta(t)$ using both methods provide accurate enough results that the overall shape of the transmission rate is easily recognized. The SI method does estimate $\beta(t)$ better in this case.

S8.2 Sensitivity to observation error in estimating $\beta(t)$ for smallpox parameters

In the main text, Figure 6 gives us a picture of the performance of the S^+ and SI methods in the presence of observation error for measles parameters. Here we will produce the same figure, but for smallpox parameters instead.

```
RR <- seq(2, 30, by = 1)
amp <- seq(0, 0.1, length.out = length(RR))

param.small.cf <- param.small
param.small.cf$cf.RR <- 0.2 # 20 % of people that get measles die from it.

beta.2D(R0.range = RR,
        amplitude.range = amp,
        param = param.small.cf,
        period = 1, noise.percent = 0,
        matrix.file.name = "Smallpox-bin-20percentCF",
        binom.dist = TRUE)
```



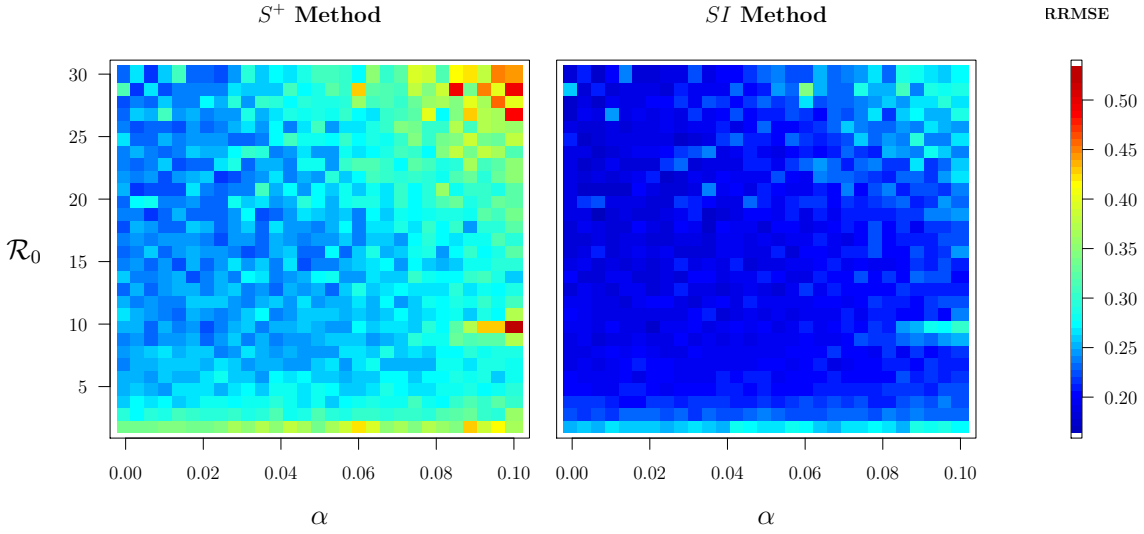
For smallpox parameters, we do see a slightly different pattern in α and $\mathcal{R}_0$ than for measles parameters. Both parameter sets indicate that small values of $\mathcal{R}_0$ ($\mathcal{R}_0 < 4$) decreases estimation of the transmission rate. However, unlike measles parameters, increasing $\mathcal{R}_0$ and α with smallpox parameters does not increase the estimation error. There is very little change in error for smallpox parameters for this range of $\mathcal{R}_0$ and α values.

S8.3 Estimation error for a range of reporting/case fatality ratio values

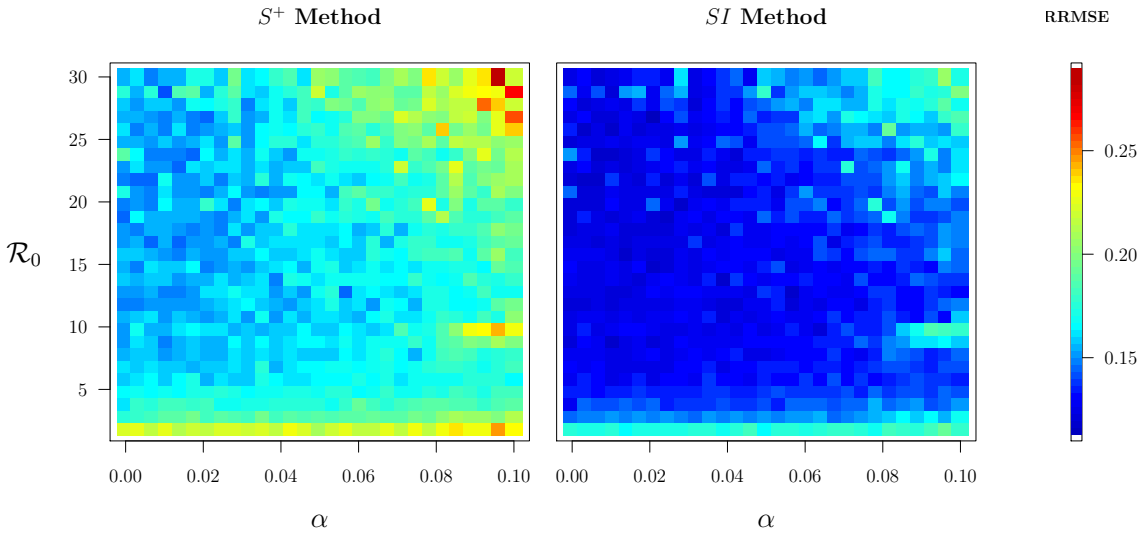
Figure 6 of the main text looks at the case of observation error when the reporting/case fatality ratio $\rho\eta$ is 20%. For $\rho\eta$ from 10% to 100%, the dependence of estimation accuracy on $\mathcal{R}_0$ and α looks qualitatively identical to Figure 6. As $\rho\eta$ increases, the observation error decreases, and so the overall estimation error is smaller. In order to demonstrate that this estimation error has the same pattern in $\mathcal{R}_0$ and α for a range of $\rho\eta$, we show a series of

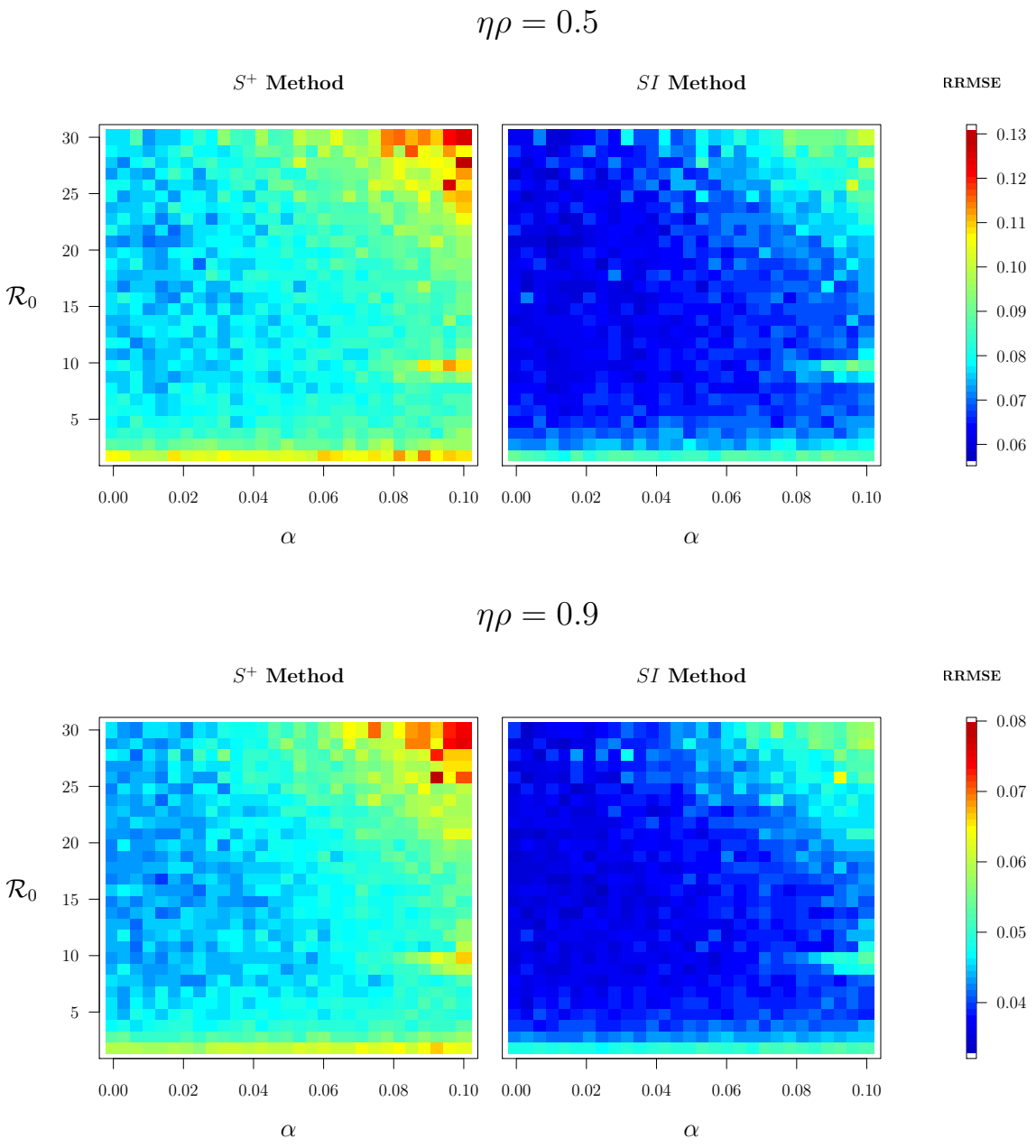
plots for $\eta\rho = 0.1, 0.2, 0.5, \& \ 0.9$ with measles parameters (plotting code suppressed).

$\eta\rho = 0.1$



$\eta\rho = 0.2$





S8.4 `R Code`: The dependence of estimation error on $\rho\eta$

The above plots for $\rho\eta = 0.1, 0.2, 0.5, 0.9$ are qualitatively very similar, however the maximum and minimum values on the RRMSE legend decrease as $\rho\eta$ increase. In order to see how this estimation error changes as a function of $\rho\eta$ we plot the maximum and minimum estimation error for $\mathcal{R}_0 \in [0, 30]$ and $\alpha \in [0, 0.1]$ for a range of $\rho\eta$ values from 0.05 to 1. This is done with the function `max.min()`:

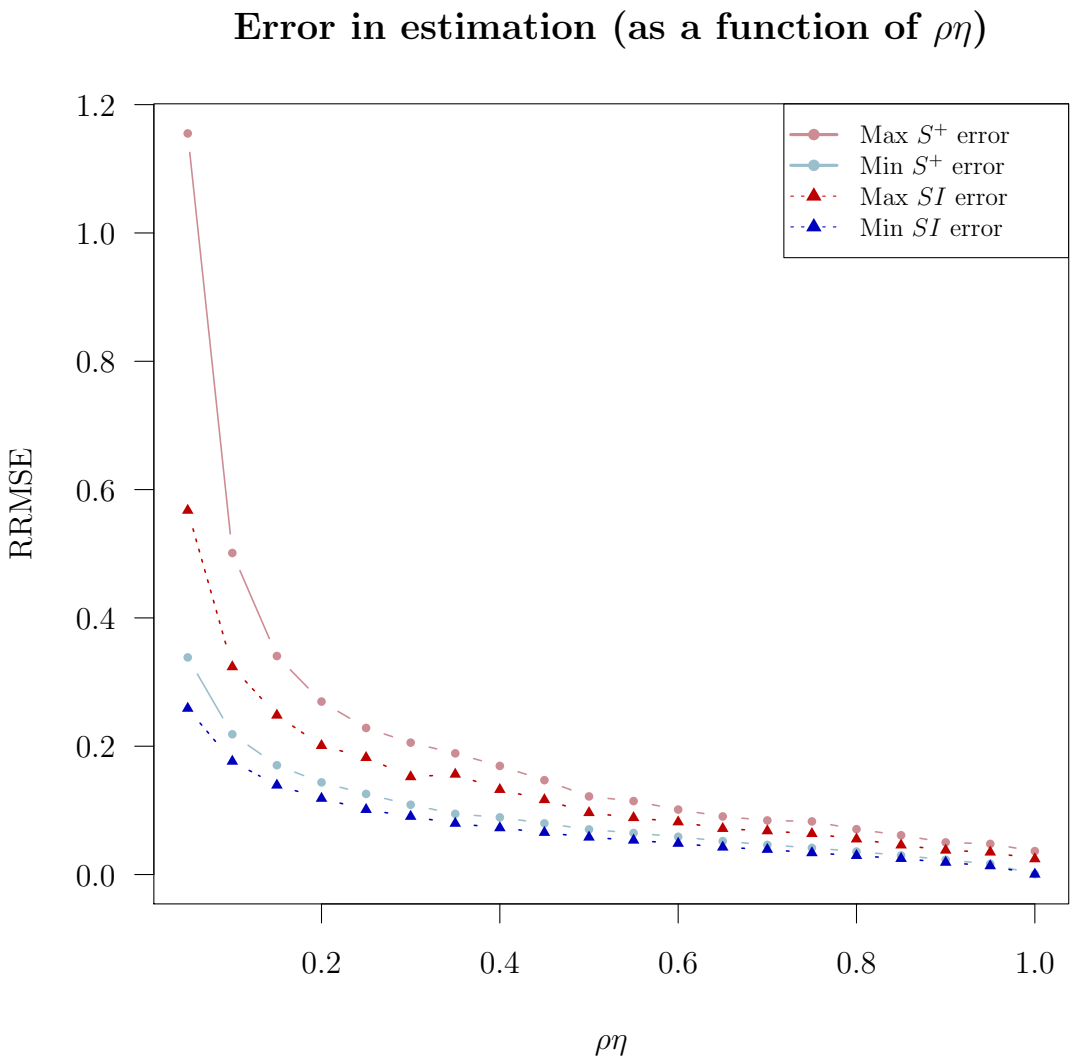
```
max.min <- function(cf.range,  # range of reporting/case fatality ratios
                    given.param){ # parameter values
  ## if we haven't already computed the max and min values
  if (!file.exists("CFlegend.Rdata")){
    ## create space for the max and min for the S+ and SI methods
    max.val.cflegend.SI <- rep(NA, length(cf.range))
    min.val.cflegend.SI <- rep(NA, length(cf.range))
    max.val.cflegend.Splus <- rep(NA, length(cf.range))
    min.val.cflegend.Splus <- rep(NA, length(cf.range))
    ## for every reporting/case fatality ratio compute the maximum
    ## and minimum estimation error
    for(index in 1:length(cf.range)){
      given.param$cf.RR <- cf.range[index]
      ## (we increase R0 by 2 to cut down on computation time)
      R0.range <- seq(2, 30, by = 2) # then for R0 from 2 to 30
      ## and alpha from 0 to 0.1:
      amplitude.range = seq(0, 0.1, length.out = length(R0.range))
      ## compute the estimation error
      error.output <- comp.error.2d(R0.range = R0.range,
                                    amplitude.range = amplitude.range,
                                    param = given.param,
                                    period = 1,
                                    noise.percent = 0,
                                    five.year.delay = FALSE,
                                    binom.dist = TRUE,
                                    matrix.file.name = "temp.csv")
      ## then store the min and max estimation error for each method
      max.val.cflegend.SI[index] <- max(error.output[[2]])
      min.val.cflegend.SI[index] <- min(error.output[[2]])
      max.val.cflegend.Splus[index] <- max(error.output[[1]])
      min.val.cflegend.Splus[index] <- min(error.output[[1]])
      ## And save these vectors in a data set
      save(min.val.cflegend.SI, max.val.cflegend.SI,
           min.val.cflegend.Splus, max.val.cflegend.Splus,
           file = "CFlegend.Rdata")
    }
  }

  ## Then load the max and min estimation error values
  load("CFlegend.Rdata")
  ## And plot the results:
  library("colorRamps")
  colour.list <-   blue2green2red(51)
  red.col <- colour.list[51] # define plotting colours
```

```
  blue.col <- colour.list[1]
  min <- min(c(min.val.cflegend.Splus, min.val.cflegend.SI))
  max <- max(c(max.val.cflegend.Splus, max.val.cflegend.SI))
  y.range <- c(min, max)
  ## plot min of S+
  plot(cf.range, min.val.cflegend.Splus, type = "b",
       lwd = 1.5, pch = 16, cex = 0.6,
       ylab = "RRMSE", xlab = "$\\rho \\eta$",
       ylim = y.range, col = "lightblue3", las = 1,
       main = "Error in estimation (as a function of $\\rho \\eta$)")
  ## plot max of S+
  lines(cf.range, max.val.cflegend.Splus, type = "b",
        lwd = 1.5, pch = 16, cex = 0.6, col = "lightpink3")
  ## plot min of SI
  lines(cf.range, max.val.cflegend.SI, type = "b",
        lwd = 1.5, lty = 3, pch = 17, cex = 0.6, col = red.col)
  ## plot max of SI
  lines(cf.range, min.val.cflegend.SI, type = "b",
        lwd = 1.5, lty = 3, pch = 17, cex = 0.6, col = blue.col)
  ## create a legend:
  legend("topright", c("Max $S^+$ error", "Min $S^+$ error",
                       "Max $SI$ error", "Min $SI$ error"),
         col = c("lightpink3", "lightblue3", red.col, blue.col),
         cex = 0.8, lty = c(1,1, 3, 3),
         lwd = c(3, 3, 1, 1), pch = c(16, 16, 17, 17))
}
```

S8.4.1 Maximum and minimum $\beta(t)$-estimation error for $\rho\eta \in [0.05, 1]$

Now lets actually use `max.min()` to look at the maximum and minimum estimation error for $\mathcal{R}_0 \in [2, 30], \alpha \in [0, 0.1]$ for a range of $\rho\eta$.

```
## for a range of reporting/case fatality ratios:
cf.range <- seq(0.05, 1, by = 0.05)
## find the max and min estimation error and plot it
max.min(cf.range = cf.range, given.param = param.meas)
```

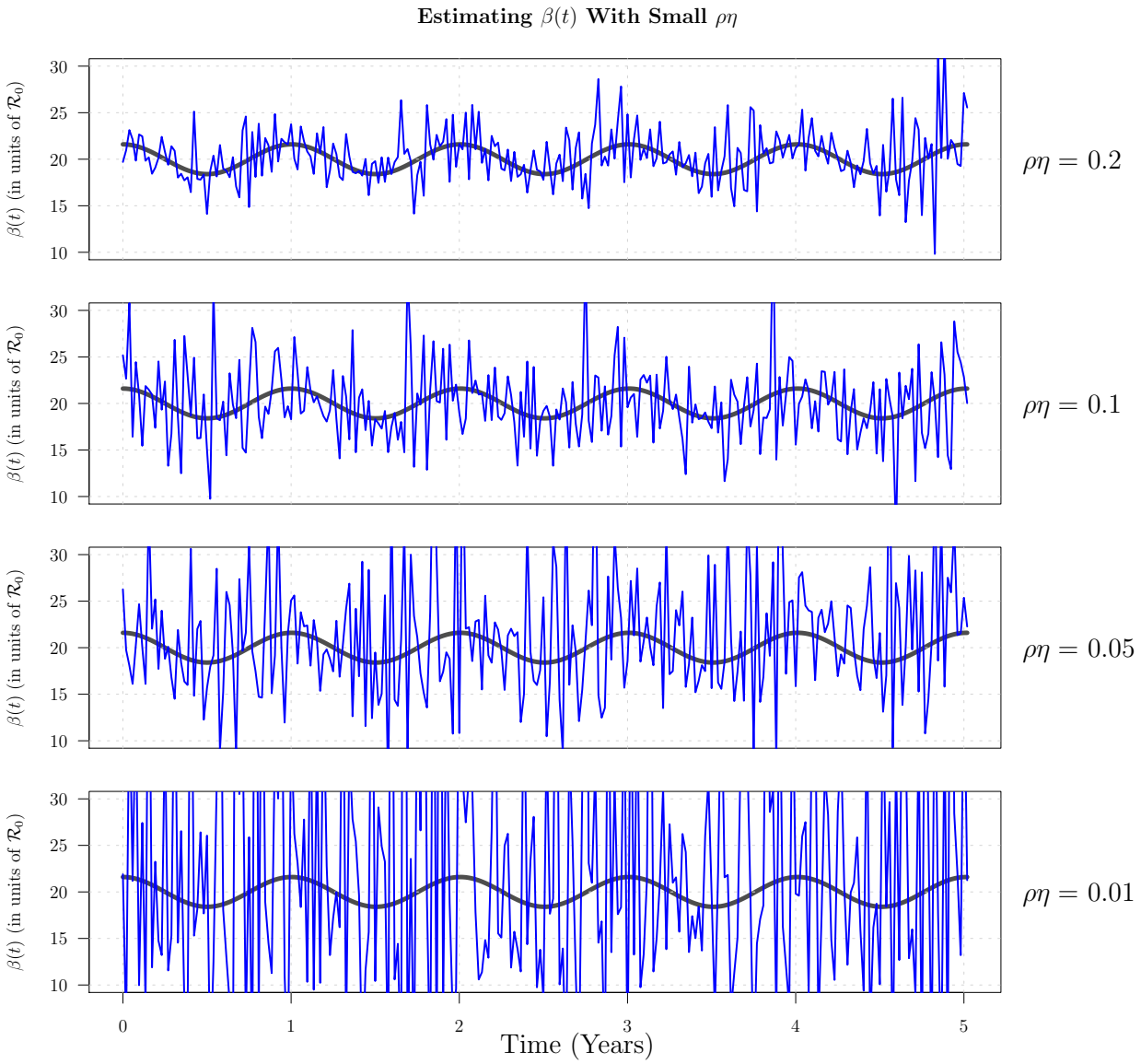


S8.5 Estimating the transmission rate with very small reporting/case fatality ratios

Very small reporting/case fatality ratios make estimation of the transmission rate very inaccurate, to the point where the estimated $\beta(t)$ is not helpful in determining any characteristics of $\beta(t)$. Here we demonstrate estimation of $\beta(t)$ using the SI method, for measles parameters with $\rho\eta = 0.2, 0.1, 0.05$, and 0.01. As $\rho\eta$ decreases, the estimate for $\beta(t)$ grows increasingly noisy.

```
## define the measles parameter set
meas.base <- param.define(type = "Measles", no.years = 20)
## define beta(t)
beta <- Create.Beta(param.list = meas.base, amp = 0.08,
```

```
                    period = 1, noise.percent = 0)
## Estimate beta(t) with four different case fatality ratios:
par(mfrow = c(4,1), mar = c(2, 4, 0, 7), oma = c(4, 0, 4, 0))
for (cf in c(0.2, 0.1, 0.05, 0.01)){
  ## define the parameter set:
  cfparam <- meas.base
  cfparam$cf.RR <- cf
  ## generate simulated case notification data
  CaseNote <- solve.SIR(cfparam, beta = beta, binom.dist = TRUE)
  ## estimate beta(t) with the SI method
  Est.cf <- Estimate.Beta(append(cfparam, list(C = CaseNote$C)))[,6]
  end <- which(cfparam$times > 52*5)[1] # only plot the first 5 years
  # we will plot beta in units of R0, so define a multiplier (mult)
  mult <- cfparam$pop.size/(cfparam$gamma.val + cfparam$mu)
  max.val <- mult*max(beta)*2 # max plotting value
  min.val <- mult*min(beta)*0.5 # min plotting value
  cf.title <- paste("$\\rho \\eta = $", cf)
  ## plot the true beta(t)
  plot(cfparam$times[1:end]/52, mult*beta[1:end],
       # ylim = c(min.val, max.val),
       ylim = c(10, 30),
       lwd = 5, type = "l", xaxt = "n", xlab = "",
       ylab = "$\\beta(t)$ (in units of $\\R$)",
       col = gray(0.3), las = 1)
  if (cf == 0.01){ # if this is that last plot
    ## add an x-axis:
    axis(1)
    mtext("Time (Years)", side = 1, outer = TRUE)
  }
  if (cf == 0.2){ # if this is the first plot, add a title
    title("Estimating $\\beta(t)$ With Small $\\rho \\eta$", outer = TRUE)
  }
  grid()
  ## plot the estimate
  lines(cfparam$times[1:end]/52, mult*Est.cf[1:end], col = "blue", lwd = 2)
  mtext(cf.title, side = 4, line = 1, las = 1)
} # end for loop
```



S9 Sensitivity to process error

In order to incorporate discreteness and demographic stochasticity into the simulation of case notification data, we use the Gillespie algorithm [2] to provide realizations of the stochastic SEIR model. Using measles parameters, for each value of $\mathcal{R}_0$ and α, we ran 100 realizations of the stochastic SEIR model and then used the data from these realizations to estimate $\beta(t)$. For each estimated $\beta(t)$ we computed the estimation error (RRMSE) and for every $\mathcal{R}_0$-α pair we took the median estimation error over the 100 realizations. We looked at population sizes of $N_0 = 100,000, 500,000$, and 1 million.

S9.1 `R Code`: Sensitivity to process error

For each $\mathcal{R}_0$-α pair ($\mathcal{R}_0 \in (2, 4, 8, 16, 32)$ and $\alpha \in (0, 0.025, 0.05, 0.075, 0.1)$), we ran 100 realizations of the stochastic SEIR model to produce 100 time series of simulated case notification data. Then $\beta(t)$ was estimated using the simulated data, and the median estimation error over the 100 data sets was recorded. `Est.With.Process.Error()` computes and plots the median estimation error for a given range of $\mathcal{R}_0$ and α.

```
Est.With.Process.Error <- function(R0.sequence, # R0 values
                                   alpha.sequence, # alpha values
                                   population.size,
                                   stored.data.name,
                                   outer.title = ""){
  ## stored.data name.contains the median error in estimating
  ## the transmission rate, beta(t) using the S+ and SI methods
  ## across 100 realizations of the stochastic SEIR model.

  ## if stored.data.name does not exist, compute the error in estimation
  if (!file.exists(stored.data.name)){

    ## create a data.list that store the names of the
    ## data set over which we want to look
    data.list <- rep(NA, length(R0.sequence))

    ## population = pop.mult*10^pop.exp for data file notation
    pop.exp <- floor(log10(population.size))
    pop.mult <- population.size/(10^pop.exp)

    for (index in 1:length(R0.sequence)){ # then for each R0 value
      ## multiply alpha value by 100 for data file notation

      alpha.val <- toString(100*alpha.sequence[index])
      ## There is a bit of a hiccup in the naming of the.csv files
      ## where alpha = 02 in the .csv file name, means actually 025
      ## and alpha = 08 in the .csv file name,  means actually 075.
      ## so let's fix the adjustment of our naming system in these two cases.
      if (alpha.sequence[index] == 0.025){
        alpha.val <- toString(100*0.02)
      }
      else if (alpha.sequence[index] == 0.075){
        alpha.val <- toString(100*0.08)
      }

      ## Then create the parts of the string to identify
      ## the data file we want
```

```
    if ((100*alpha.sequence[index]/10) < 1){ # need 2 digits for alpha
      alpha.val <- paste("0", alpha.val, sep = "")
    }
    R0.val <- toString(R0.sequence[index])
    if (R0.sequence[index]/10 < 1){ # need 2 digits for R0
      R0.val <- paste("0", R0.val, sep = "")
    }
    ## then store the data name in data.list
    data.list[index] <- paste("GillSim-R0=", R0.val, "-alpha=",
                              alpha.val, "-pop=", pop.exp,
                              "-popmult=", pop.mult, ".csv", sep = "")
  }

  ## Then for each dataset contained in data.list we will compute
  ## the median error in estimation over the 100 realizations contained
  ## in the data set.  We will also keep track of how many of the
  ## realizations fade out before the end of 20 years.

  ## create space
  error.SI <- rep(NA, length(data.list)) # error using SI method
  error.Splus <- rep(NA, length(data.list)) # error using S+ method
  doesnt.finish <- rep(NA, length(data.list)) # percent fade out

  ## Then run the function ProcessError (defined below)
  ## for each of the data sets to compute the estimation
  ## error and probability of fade out.
  index <- 1
  for (dataset in data.list){
    output <- Process.Error(dataset)
    error.Splus[index] <- output[1]
    error.SI[index] <- output[2]
    doesnt.finish[index] <- output[3]
    index <- index + 1
  }
  ## then save these elements to the stored data name.
  save(error.SI, error.Splus, doesnt.finish, file = stored.data.name)
}

## load stored.data.name and plot the process error.
load(stored.data.name)
ProcessError.Plot(error.SI = error.SI,
                  error.Splus = error.Splus,
                  y.values = R0.sequence, # R0 ranges
                  x.values = alpha.sequence, # alpha range
```

```
                    percent.fade.out = doesnt.finish,
                    outer.title = outer.title)
}
```

`Process.Error()` is the function that does the bulk of the work in computing the error in estimating $\beta(t)$. It also records the percentage of cases that end in fade out for each $\mathcal{R}_0$-α pairings.

The 100 sets of simulated data have been stored in a data file named: ***GillSim-R0=xx-alpha=yy-pop=z-popmult=w.csv***, where xx is the value for $\mathcal{R}_0$ in two digits (*e.g.*, $xx = 05$ if $\mathcal{R}_0 = 5$), yy is the value for alpha after the decimal (*e.g.*, $\alpha = 0.08 => yy = 08$), z is the exponent on ten for the population size ($N = 10^z$), and popmult is the multiple of the population size ($N = w10^z$). `Process.Error()` assumes the data set in named in this manner, and that the distance between recorded time points is equal to the time unit.

```
library(stringr)

Process.Error <- function(data.set){
  ## define the measles parameter set
  param.meas <- param.define(type = "Measles", no.years = 20)
  ## Determine R0, alpha, and the population from the data.set name
  R0 <- as.numeric(str_sub(data.set, start = 12, end = 13))
  alpha <- as.numeric(str_sub(data.set, start = 21, end = 22))/100
  pop.exponent <- as.numeric(str_sub(data.set, start = 28, end = 28))
  pop.mult <- as.numeric(str_sub(data.set, start = 38, end = 39))
  pop.size <- pop.mult*(10^pop.exponent)

  ## fix alpha (we should have saved the data sets
  ## with three digits of accuracy for alpha, but since we didn't
  ## those with three digits need to be corrected)
  if (alpha == 0.02){
    alpha <- 0.025
  }
  else if (alpha == 0.08){
    alpha <- 0.075
  }
  Birth.input <- rep(round(pop.size*param.meas$mu), length(param.meas$times))

  ## define the initial conditions:
  mean.beta <- (R0*(param.meas$gamma.val + param.meas$mu)/pop.size)
  Init.S <- 1/R0
  Init.I <- (R0 - 1)*param.meas$mu/(mean.beta*pop.size)
  Init.R <- 1 - Init.S - Init.I
```

```
## Then generate a parameter list to use from now on

param.list <- list(times = param.meas$times,
                   pop.size = pop.size,
                   gamma.val = param.meas$gamma.val,
                   mu = param.meas$mu,
                   ## in the stochastic realizations no delay in reporting
                   ## so t.report = 0
                   t.report = 0,
                   t.recover = param.meas$t.recover,
                   cf.RR = param.meas$cf.RR,
                   R0 = R0,
                   Birth.input = Birth.input,
                   Init.S = Init.S,
                   Init.I = Init.I,
                   Init.R = Init.R)

## Then generate the true beta_t that we are using in this case:
beta <- Create.Beta(param.list = param.list, amp = alpha,
                    period = 1, noise.percent = 0)

## Read in the data generated from the stochastic SEIR model
name.in <- paste("GS/", data.set, sep = "") # In a folder:GS
data.in <- read.csv(name.in)
## count the number of realizations (100 in our case)
no.realizations <- dim(data.in)[2] - 2 #the first 2 cols are for time

## set up space for the following vectors:
## Error in beta estimation for each of the realizations
S.plus.error <- rep(NA, no.realizations)  # S+ method
SI.error <- rep(NA, no.realizations)  # SI method
## Number of realizations that fade out before the 20 years
doesnt.finish <- 0

## Then for each of the columns that record a set of simulated
## case notification data, lets estimate beta and compute the error
## in our calculation.
for (column.index in 3:(no.realizations + 2)){
  ## Compute beta using S+ and SI Method:
  extended.params <- append(param.list, list(C = data.in[, column.index]))
  Estimates <- Estimate.Beta(extended.params)

  ## If there is fade out, we want the beta estimate to be NA and not 0.
  ## This already happens in the S+ method (since we get divide by zero)
```

```
    ## But needs to be adjusted for the SI method (stored in Estimates[,6])
    no.zeros <- which(Estimates[,6] == 0)
    Estimates[,6][no.zeros] <- rep(NA, length(no.zeros))

    ## then compute the error in estimation
    mean.val <- mean(beta, na.rm = TRUE)
    no.points <- dim(data.in)[1]
    no.NA.vals <- length(which(is.na(Estimates[,1])))
    real.no.points <- no.points - no.NA.vals
    div <- sqrt(real.no.points)*mean.val
    S.plus.error[(column.index - 2)] <- e.dist(Estimates[,1], beta)/div
    SI.error[(column.index - 2)] <- e.dist(Estimates[,6], beta)/div

    ## Check if there was fade out before 20 years.
    ## We generally have a lot of error during the time around fade out.
    if (is.na(Estimates[(no.points-10), 6])){
      doesnt.finish <- doesnt.finish + 1
    }
  } # end loop over all the realizations

  ## then take the median of all the errors
  median.S.plus.error <- median(S.plus.error, na.rm = TRUE)
  median.SI.error <- median(SI.error, na.rm = TRUE)

  return(c(median.S.plus.error, median.SI.error, doesnt.finish))
}
```

457 `ProcessError.Plot` then plots the error in estimation of $\beta(t)$ for the S^+ and SI method,
458 as well as the probability of fade out before the end of 20 years.

```
ProcessError.Plot <- function(error.SI, # estimation error (SI method)
                              error.Splus, # estimation error (S+ method)
                              y.values, # R0 values
                              x.values, # alpha values
                              # percent cases that fade out before 20 yrs
                              percent.fade.out, outer.title = ""){
  ## define plotting layout and parameters
  par(oma = c(0,5,3,0))
  layout(matrix(c(1, 1, 2, 2, 3, 4, 4, 5), nrow = 1, ncol = 8, byrow = TRUE))
  cex.val <- 9 # size of the squares plotted
  ## define the colour palatte:
  library("colorRamps")
  part.c <- blue2green2red(51)
  part.a <- rev(gray(seq(0.1, 0.98, length.out = 15)))
```

```
part.b <- colorRampPalette(c("gray20", part.c[1]))(3)
list.of.colors <- c(part.a, part.b, part.c)

## 1. plot S+ method estimation error
par(mar = c(5, 0.1, 3, 0.1))
plot(0, 0, type = "l", col = "white", ylim = c(log(1.4), log(45)),
     xlim = c(min(x.values - 0.01), max(x.values + 0.01)),
     xlab = "$\\alpha$", ylab = "", cex.lab = 1.5,
     main = "S+ Method", yaxt = "n")
axis(side = 2, at = log(c(2, 4, 8, 16, 32)),
     labels = c(2, 4, 8, 16, 32), las = 1)
mtext("$\\R$", side = 2, line = 3, las = 1)
for (index in 1:length(error.Splus)){
  val <- error.Splus[index]
  p <- calc.col2(val, list.of.colors) # determine colour
  points(x.values[index], log(y.values[index]), col = p,
         pch = 15, cex = cex.val)
}

## 2. plot SI method estimation error
par(mar = c(5, 0.1, 3, 0.1))
plot(0, 0, type = "l", col = "white", ylim = c(log(1.4), log(45)),
     xlim = c(min(x.values - 0.01), max(x.values + 0.01)),
     xlab = "$\\alpha$",  ylab = "$\\R$", main = "SI Method",
     yaxt = "n", cex.lab = 1.5)
for (index in 1:length(error.SI)){
  val <- error.SI[index]
  p <- calc.col2(val, list.of.colors) # determine colour
  points(x.values[index], log(y.values[index]), col = p,
         pch = 15, cex = cex.val)
}

## 3. create a color legend on the side for error estimation
par(mar = c(5, 0.5, 3, 5))
list.of.vals <- c(seq(0.04, 0.1, length.out = (length(list.of.colors)/2)),
                  seq(0.1, 0.45, length.out = length(list.of.colors)/2))
plot(0, 0, col = "white", ylim = log(c(0.04, 0.45)),
     xlab = "", xaxt = "n",
     ylab = "", yaxt = "n")
axis.vec <- c(0.04, 0.06, 0.08, 0.1, 0.2, 0.3, 0.4, 0.5)
axis(4, at = log(axis.vec), labels = axis.vec, las = 1)
for (index in 1:length(list.of.vals)){
  points(0, log(list.of.vals[index]), col = list.of.colors[index],
         pch = 15, cex = 2)
```

```
  }
  mtext("RRMSE", side = 3, line = 1, cex = 0.8)


  ## 4. plot the percent of case that fade out before 20 years
  par(mar = c(5, 0.5, 3, 0.1))
  ## define a new colour scheme,
  fadeout.col.scheme <- c("gray95", brewer.pal(9, "YlOrRd"))
  fadeout.vals <- seq(0, 100, length.out = length(fadeout.col.scheme))
  plot(0, 0, col = "white", ylim = c(log(1.4), log(45)),
       xlim = c(min(x.values - 0.01), max(x.values + 0.01)), yaxt = "n",
       ylab = "", xlab = "$\\alpha$", cex.lab = 1.5,
       main = "Probability of Fadeout")
  axis(side = 2, at = log(c(2, 4, 8, 16, 32)),
       labels = c(2, 4, 8, 16, 32), las = 1)
  for (index in 1:length(percent.fade.out)){
    val <- percent.fade.out[index]
    p <- fadeout.col.scheme[(which(fadeout.vals >= val)[1])]
    points(x.values[index], log(y.values[index]), col = p,
           pch = 15, cex = cex.val)
  }

  ## 5. add legend for fadeout percentage
  par(mar = c(5, 0.5, 3, 5))
  plot(0, 0, col = "white", ylim = c(min(fadeout.vals), max(fadeout.vals)),
       xlab = "", xaxt = "n",
       ylab = "", yaxt = "n")
  axis(4, las = 1, at = seq(0, 100, by = 10),
       labels = paste(seq(0, 100, by = 10), "$\\%$", sep = ""))
  for (index in 1:length(fadeout.vals)){
    points(0, fadeout.vals[index], col = fadeout.col.scheme[index],
           pch = 15, cex = 5.5)
  }

  ## If outer.title is specified, list the title:
  if (!(outer.title == "")){
    mtext(outer.title, side = 3, outer = TRUE)
  }
}
```

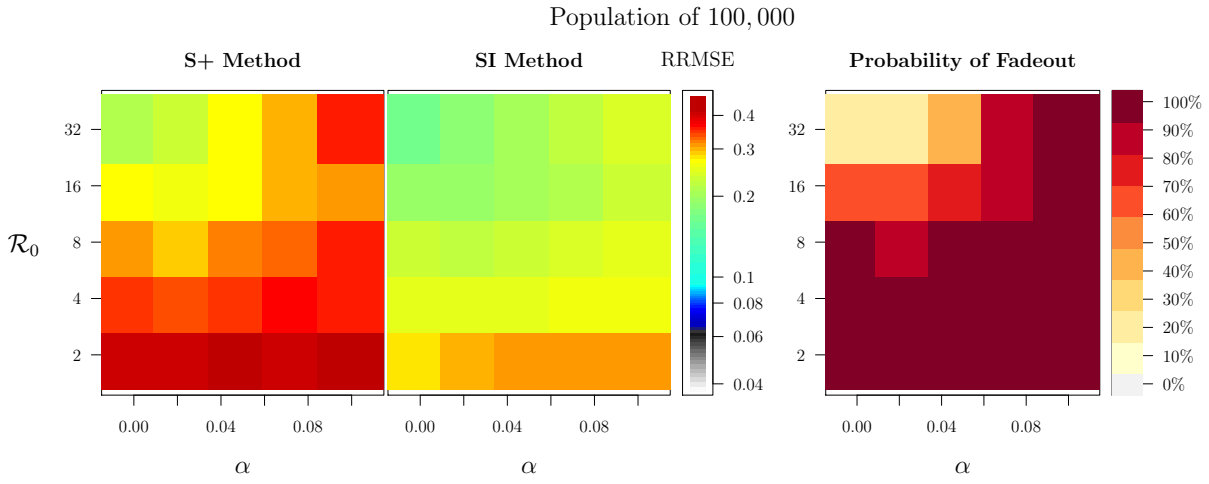
```
## calc.col2 determines which colour to plot each point.
calc.col2 <- function(val, list.of.colors){
  list.of.vals <- c(seq(0.04, 0.1, length.out = (length(list.of.colors)/2)),
```

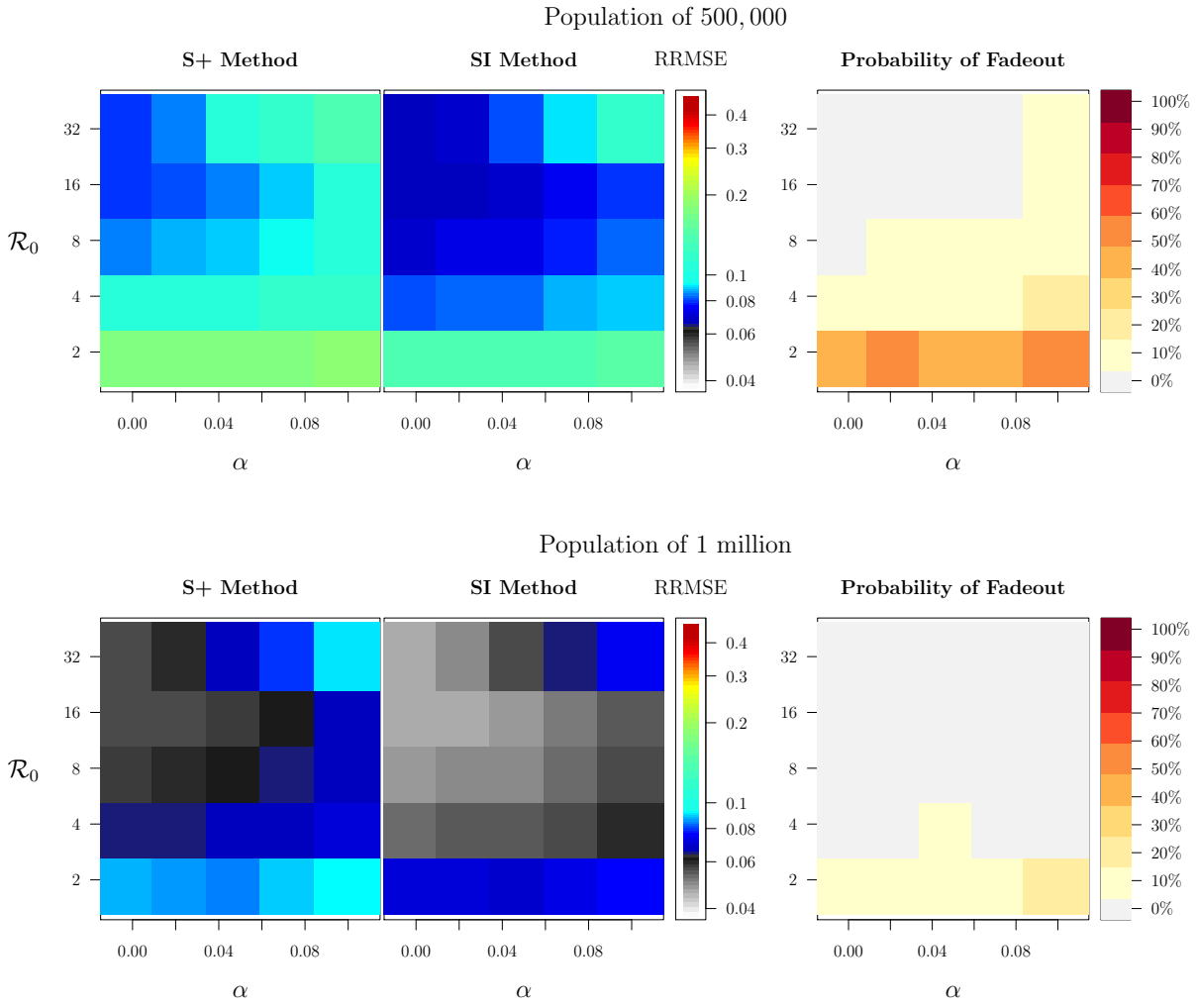
```
                    seq(0.1, 0.45, length.out = length(list.of.colors)/2))
  k <- which(list.of.vals >= val)[1]
  plot.col <- list.of.colors[k]
  return(plot.col)
}
```

S9.2 Comparing estimation error for a range of population sizes

In the main text, Figures 8–9 display the $\beta(t)$-estimation error for a range of $\mathcal{R}_0$ and α values with a population of $100,000$ and 1 million in the presence of process error. Here we will also include the estimation error for a population size of $500,000$. We will display the R code used to compute and plot the estimation error for a population size of $100,000$. The R code for the other population sizes is suppressed as it is almost identical.

```
## Case 1:  Population size of 100,000
R0.sequence <- c(rep(2, 5), rep(4,5), rep(8, 5), rep(16, 5), rep(32,5))
alpha.sequence <- rep(c(0, 0.025, 0.05, 0.075, 0.1), 5)
output.pop1 <- Est.With.Process.Error(R0.sequence = R0.sequence,
                                      alpha.sequence = alpha.sequence,
                                      population.size = 100000,
                                      stored.data.name = "GSError1.Rdata",
                                      outer.title =
                                        "Population of $100,000$")
```

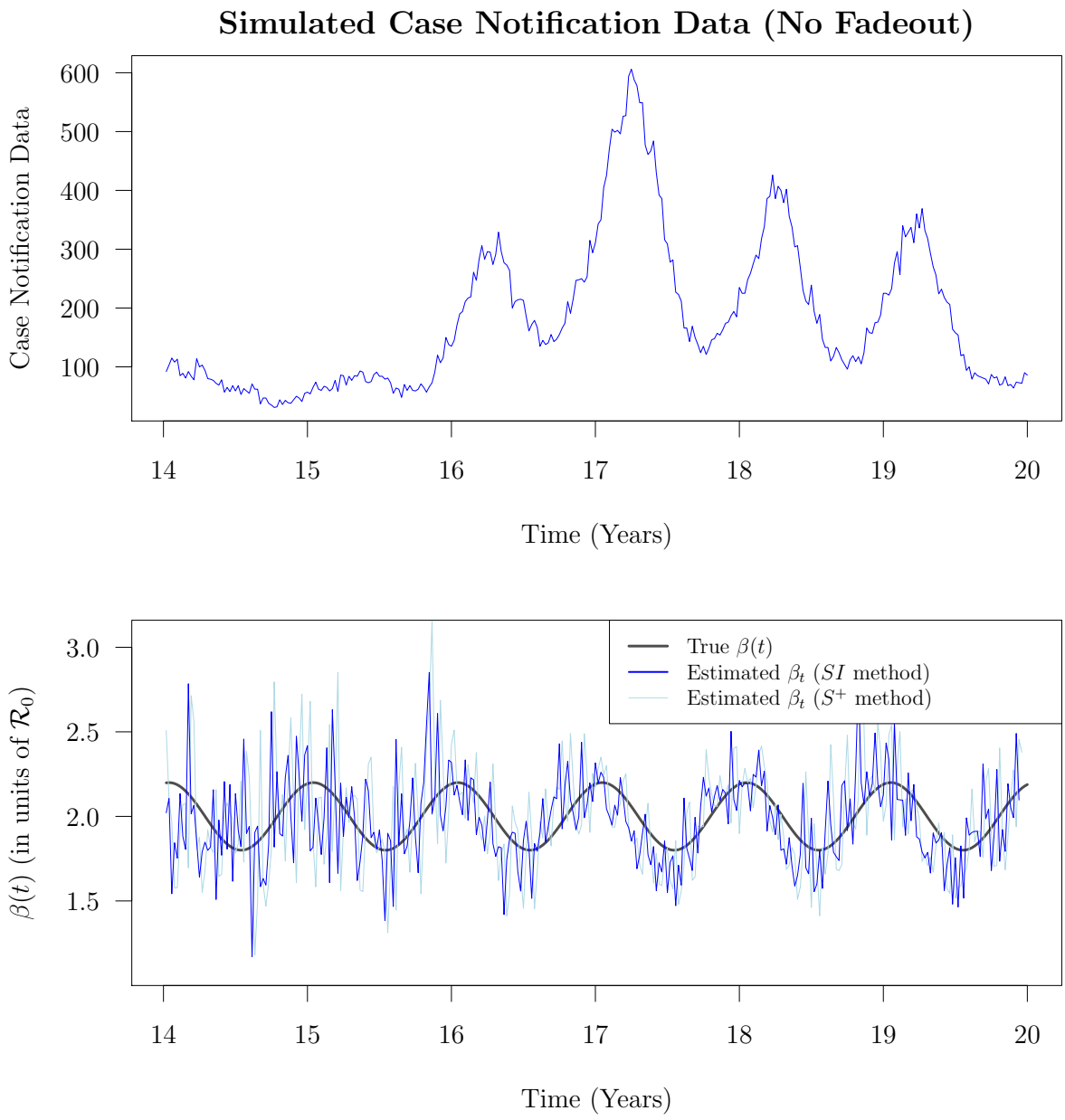


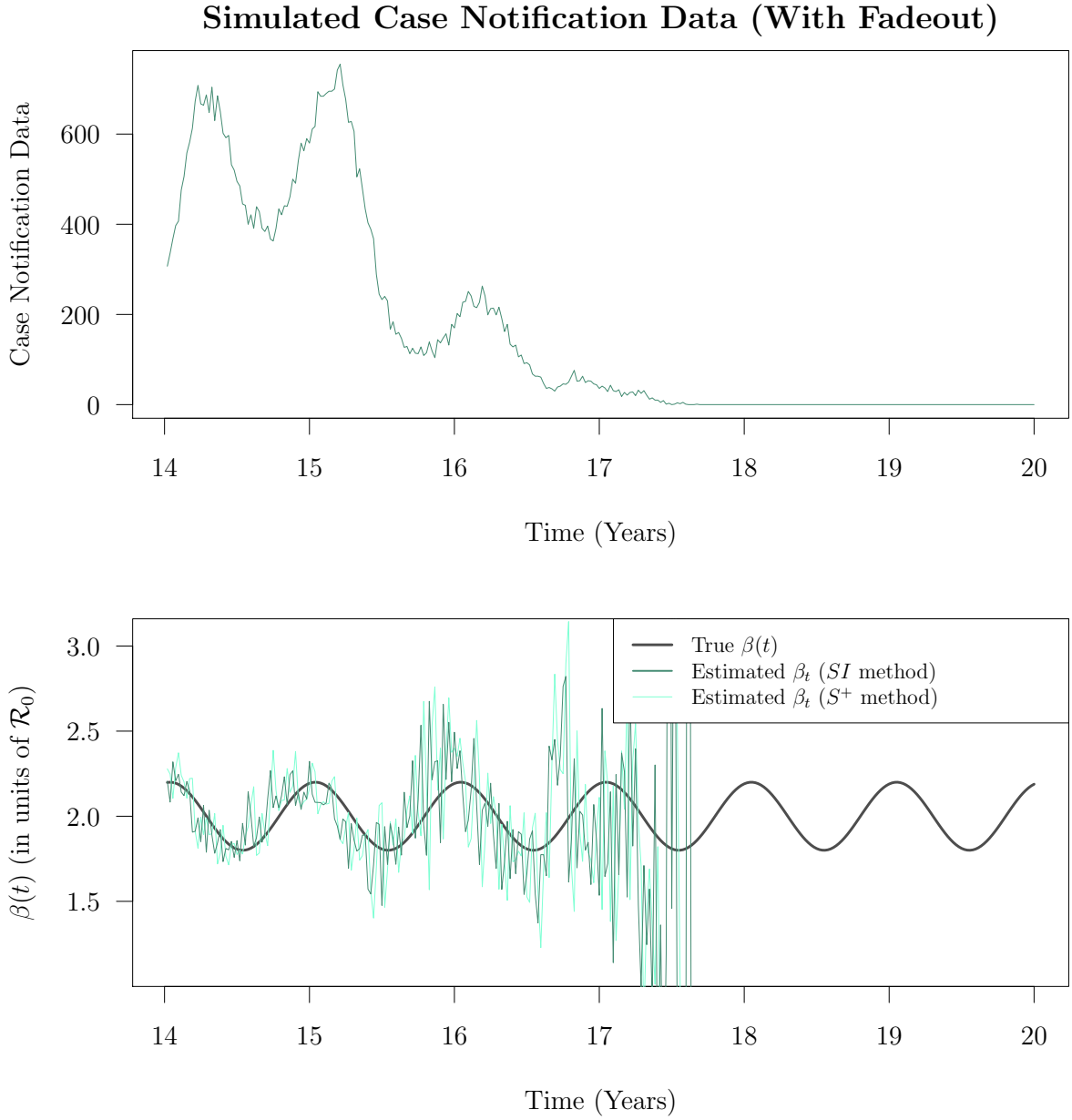


Increasing the population size increases estimation accuracy in both the S^+ and SI method. Similar to estimates with observation error, small values of $\mathcal{R}_0$ and large values of $\mathcal{R}_0$ and α have greater estimation error. Estimating $\beta(t)$ is inaccurate right before the fade out of the disease (if it fades out). This is why estimation error for $\mathcal{R}_0$-α pairs where there is a high probability of fadeout is much higher than cases with a low probability of fadeout.

S9.3 $\beta(t)$-estimation for an infectious disease that fades out

The estimation methods are inaccurate right before the disease fades out. To see an example of this, let's look at two estimates of $\beta(t)$, one in the case when the disease fades out, and one in the case when the disease persists. We will use measles parameters, but with $\mathcal{R}_0 = 2, \alpha = 0.1$, and $N = 500,000$.


Simulated Case Notification Data (No Fadeout)
Case Notification Data
600
500
400
300
200
100
14
15
16
17
18
19
20
Time (Years)
$\beta(t)$ (in units of $\mathcal{R}_0$)
3.0
2.5
2.0
1.5
True $\beta(t)$
Estimated β_t (SI method)
Estimated β_t (S^+ method)
14
15
16
17
18
19
20
Time (Years)

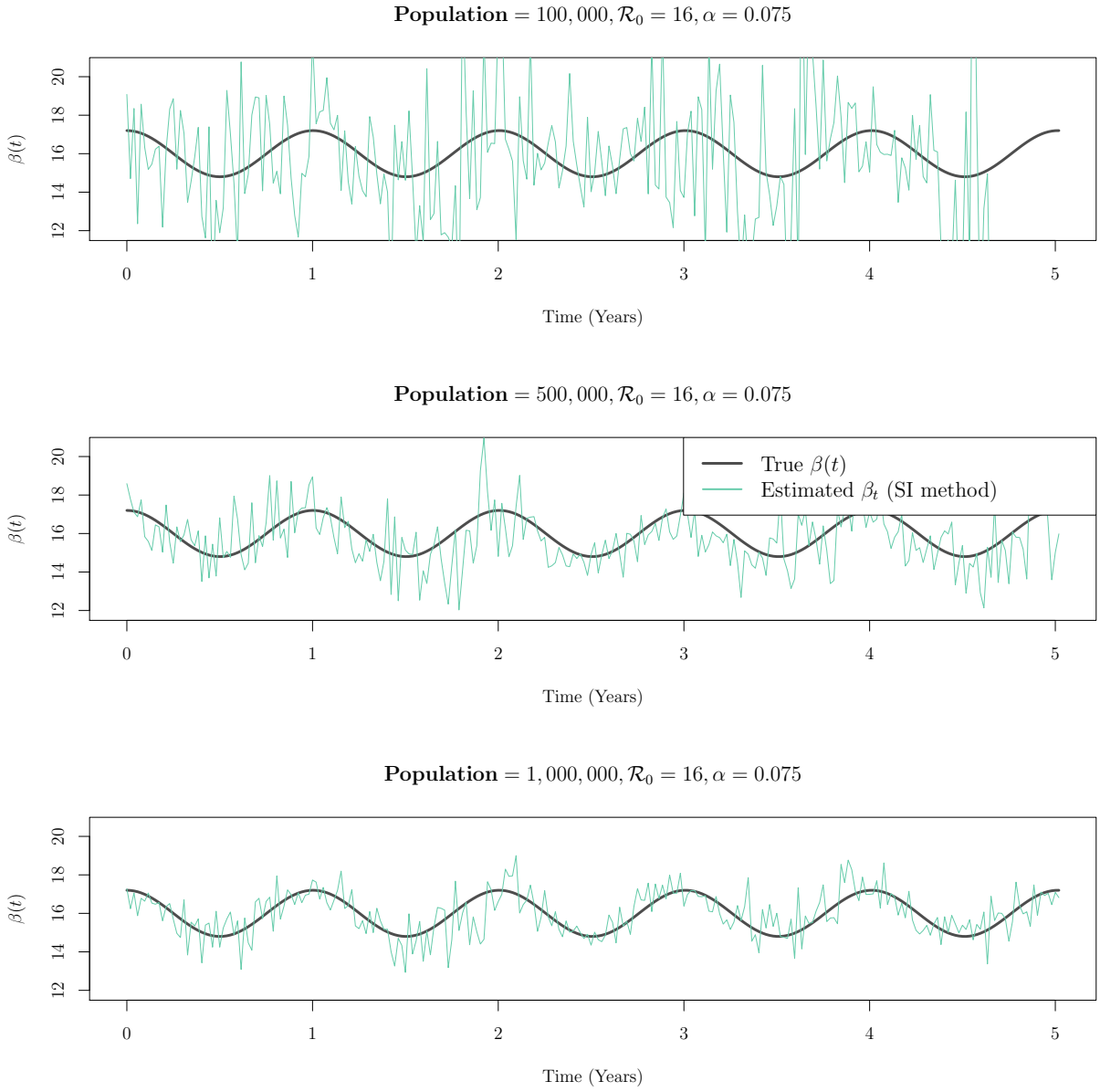


These figures make it clear that both the S^+ and SI estimation methods are unable to estimate $\beta(t)$ well right before fadeout of the disease, resulting in an estimate that oscillating wildly. We also notice that smaller amounts of case notifications each time step results in a noisier estimate of $\beta(t)$. This is especially clear in the case without fadeout when comparing years 14 and 15 with years $17 - 20$.

S9.4 Plotting the estimate $\beta(t)$ for a range of population sizes

Although the figures in §S9.2 give us an idea of the estimation accuracy for a range of $\mathcal{R}_0$, α, and population sizes, it is often helpful to have an idea of what an estimate for $\beta(t)$ with that sort of error looks like. With measles parameters we plot the true and estimated $\beta(t)$ in the presence of process error, for a population of $100,000$, $500,000$ and $1,000,000$. As

we increase the population size, the estimate of $\beta(t)$ becomes less noisy, and the underlying transmission rate $\beta(t)$ becomes much more apparent.



References

[1] D. J. D. Earn, P. Rohani, B. M. Bolker, and B. T. Grenfell. A simple model for complex dynamical transitions in epidemics. *Science*, 287(5453):667–670, 2000.

[2] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.

[3] O Krylova. *Predicting epidemiological transitions in infectious disease dynamics. Smallpox in historic London (1664 - 1930).* PhD thesis, McMaster University, Canada, 2011.

[4] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2012. ISBN 3-900051-07-0.

[5] Yihui Xie. *knitr: A general-purpose package for dynamic report generation in R*, 2013. R package version 1.1.