BAYESIAN FORECASTING OF INFECTIOUS DISEASE EPIDEMICS

Édouard F. Bonneville (s1914944)

External supervisor: Prof. Dr. J. Wallinga
(LUMC & RIVM)

First supervisor: Dr. M. Fiocco
(LUMC & LU)

MASTER'S THESIS

Defended on April 5th, 2019

Specialisation: Statistical Science

STATISTICAL SCIENCE
FOR THE LIFE AND BEHAVIOURAL SCIENCES

Abstract

Reliable forecasting of infectious disease epidemics is critical for decision-making regarding the allocation of public health resources. Until now, efforts have mostly been devoted to understanding disease transmission rather than forecasting.

The present thesis took a Bayesian approach to forecasting epidemics, focusing on modelling the time between successive observed infection events using a Gamma generalised linear model (GLM). Specifically, a tailored sampler was introduced to solve common convergence problems associated with the use of the inverse link function. The posterior distribution obtained from the Bayesian Gamma GLM was then used to forecast stochastically. This approach was extended to diseases with different transmission dynamics, as described by traditional compartmental epidemiological models: susceptible-infectious (SI), susceptible-infectious-susceptible (SIS) and susceptible-infectious-recovered (SIR). The calibration of the forecasting technique was evaluated using probability integral transform (PIT) histograms in a large simulation study.

Results showed that forecasts of SI and SIS-type epidemics in an early growth phase underestimated true future values. Across epidemic types, there was evidence of overdispersion in the forecasts. Furthermore, the method was applied to data from the Meningococcal disease, serogroup W outbreak in the Netherlands between 2012 and 2018. Forecasts suggested the outbreak has reached an equilibrium of approximately 50-55 new observed cases per 6 months.

Avenues for future research are provided, with a focus on how we could improve the Bayesian approach and adapt the method to account for covariates of interest.

Acknowledgements

I would like to express my gratitude to both of my supervisors Prof. Dr. Jacco Wallinga and Dr. Marta Fiocco, for their unending guidance, motivation and support throughout this project.

Jacco, it has been an absolute pleasure working together for the past months. I learned a great deal from the meetings we had, and continue to be inspired by the dedication and humility with which you work.

Marta, beyond your supervision I am especially grateful for your qualities as a study advisor, which were invaluable in helping me make the right decisions at the right time during the master's program.

I would also like to thank the modelling team at the Dutch National Institute for Public Health and the Environment (RIVM) for providing the Meningococcal-W dataset, and for their insights on the forecasting method.

A special thanks to Sarah von Grebmer zu Wolfsthurn for her incredible support from start to finish of this project, not least of which involved proofreading. You were uplifting at the times I needed it most, and for that, I cannot thank you enough.

To Niek Mereu; I am grateful for all the dinners, laughs, cries and most importantly conversations unrelated to statistics we have shared over the past couple of years. It has been a pleasure completing this master's with you.

To my parents and little brother; the love and encouragement you have given me has been and continues to be phenomenal. None of this would have been possible without you. Thank you for everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Epidemic forecasting

Forecasting of infectious disease epidemics is of considerable public health relevance [Gandon et al., 2016]. Early and reliable forecasts of future incidence are crucial for deciding whether or not to introduce a vaccine, and subsequently prevent future morbidity and mortality. Until recently, infectious disease modelling has been concerned with understanding and explaining the mechanisms behind disease transmission, rather than future forecasting of key epidemiological quantities [Held et al., 2017]. There seems to be an inherent difficulty in predicting the future course of epidemics, because models need to attempt to take in to account spatiotemporal dependencies (i.e. human travel over time) and contact patterns between and within different age-groups, among other factors.

Since the Ebola outbreak in 2014, interest in epidemic forecasting has surged. For example, some publications used exponential growth models to predict future number of infections [Ebola Response Team, 2014, Frieden et al., 2014]. When extrapolated over longer periods of time, these kind of models can lead to unrealistically high predicted number of infections. In addition to Ebola, there has been a longstanding interest in forecasting influenza epidemics: Nsoesie et al. [2014] discuss the various attempted models in a review article. As each disease presents itself as a different modelling challenge, a consensus is yet to be reached as to which techniques produce the highest quality forecasts.

1.2 Aims and structure

To address this question, the present thesis builds on previous work that proposed a time-to-event approach, where the events of interest are infection events. We can then model the time between successive infection events, so-called *interevent times*, as a function of explanatory variables using a Gamma generalised linear model (GLM). In doing so, we can estimate key epidemiological quantities (e.g. contact rate, final epidemic size) and forecast future number of infections. Previous research has used Gamma GLMs in conjunction with bootstrapping to produce forecasting intervals at different points in time [Verkerk, 2018]. Due to the occasional poor convergence of the Gamma GLM, reliable bootstrap confidence intervals were difficult to obtain, and as a result some (extreme) forecasts made little biological sense. In the broader epidemiological literature, Gamma GLMs of interevent times are extremely scarce; and when used, the main purpose was not forecasting. Instead, it was to evaluate risk factors associated with changes in interevent times [Checkley et al., 2009, Bateson, 2009].

The current thesis has two main aims: taking a Bayesian approach to overcome problems with convergence of the Gamma GLM to make forecasts, and extending the interevent time approach to diseases with different transmission dynamics. A concise introduction to infectious disease modelling is provided in Chapter 2, and Chapter 3 recapitulates the theory behind Gamma GLMs, as well as describing a tailored Bayesian MCMC sampler for the current problem. Chapters 4 and 5 respectively link the Gamma GLM to traditional compartmental epidemiological models, and show how we can use the posterior distribution of regression estimates to forecast. The calibration of the proposed method is assessed in Chapter 6 using probability integral transform (PIT) histograms, and subsequently applied to a real-world dataset in Chapter 7: the Meningococcal-W outbreak in the Netherlands since 2012. Finally, findings are summarised in Chapter 8, and future avenues for research are proposed.

Chapter 2

Infectious Disease Modelling

The following chapter provides an introduction to infectious disease modelling. Section 2.1 looks at basic terminology, and Sections 2.2-2.4 discuss the basic compartmental epidemiological models.

2.1 Essential concepts

Mathematical modelling of infectious diseases has roots stretching back to the 18^{th} century. It has since been primarily concerned with understanding the mechanisms by which diseases spread and persist in a population [Keeling et al., 2008]. Such research has key bearing on public health policy, where decisions frequently need to be taken regarding current and future prevention of outbreaks.

Any discussion on infectious diseases begins with the infectious *agent*, the cause of the disease; and the *host*, the organism harboring the disease. In their landmark text, Anderson and May [1992] distinguish between microparasites and macroparasites. Microparasites generally include small agents such as bacteria, viruses and protozoa. These are characterised by small generation times (i.e. rapid multiplication), and hosts often develop immunity to the agent. Macroparasites are typically much larger in size; including nematodes, tapeworms and arthropods. Of importance here is the distribution of parasites amongst hosts: few individuals tend to have a large number of parasites, while most have comparatively few. The parasite burden within a host has a direct impact on the level of immune response.

For the following, we work in the microparasitic case. If we assume the number of parasites within the host is not essential, we can divide the population into states: such as infected and immune. This gives rise to what we refer to as *compartmental* epidemiological models that describe the transmission dynamics of a disease. Before moving forth with these, let us define some basic terminology. The main references for the remainder of this chapter are Lloyd [2007] and Keeling et al. [2008], unless otherwise specified.

2.1.1 Terminology

In an epidemic, the first infection occurs by definition at time $t = 0$, after the parasite infects the first *host.* After becoming infected, symptoms need not occur immediately. We refer to *incubation period* as the time delay between contracting the infection and onset of symptoms. Furthermore, after infection a host is not necessarily immediately infectious (able to transmit the parasite): the delay between infection and infectiousness is known as the *latent period.* The latent and incubation periods do not have to be overlapping.

If we view a single infection as an event, we refer to it specifically as an *observed infection.* An infection generally becomes observed as a result of symptom onset, and subsequent hospitalisation/medical consultation. It may also occur that an infection is not reported; for example as a result of substantial overlap of symptoms between a disease and a common ailment. In such a situation, a host may be lead to attempt to treat their illness at home rather than report to a physician. This naturally leads to *under-reporting*: only a proportion of all infections in a population are reported. Additionally, some parasites can survive and multiply on the host without infecting the host, we then say that the host *carries* the parasite. Hosts can spread the parasite to others and may not have any symptoms: we thus speak of *asymptomatic carriage.*

From a population view-point, we call *diseased* the hosts exhibiting symptoms. We can then speak of disease *prevalence* (number of diseased individuals at a specific point in time), or disease *incidence* (number of new cases over time). However, compartmental models are formulated in terms of *infectiousness.*

2.2 The basic SI model

Consider a closed population, where there is an absence of births, deaths, immigration and emigration (all demographics are ignored). In this population, individuals are either classed as *susceptible* or *infectious*. Let S and I denote the number of susceptible and infectious individuals (also referred to as infectives) respectively, and N the size of the population. The upper case S and I denote the susceptible and infectious classes. As we have a closed population, it follows that $S + I = N$. In the context of an SI (susceptible-infectious) model, as represented in Figure 2.1 below, upon infection all individuals become infectious immediately and never recover. An outbreak here begins when $I = 1$, and ends when $I = N$: when the susceptible pool is depleted.

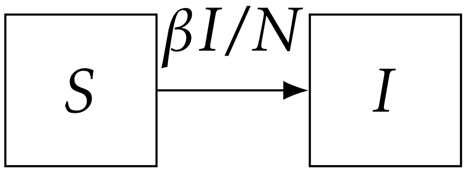

Figure 2.1: Susceptible-infectious (SI) model.

In the figure we can also observe the term $\beta I/N$: this is *force of infection* (also commonly denoted λ), which is defined as the per capita rate at which susceptible individuals become infectious. Making up the force of infection we first have I/N, which is simply the proportion of infectious individuals in the population. Second, we have the infection parameter β. This is a composite measure of contact rate and transmission probabilities. To be specific, it is the product between a) the contact rate between a susceptible individual and all other population members (e.g. someone making on average 10 contacts per day), b) the probability of disease transmission when an infectious individual contacts a susceptible individual. To make the parallel with the field of Survival Analysis, this force of infection is technically a *hazard* rate for susceptibles.

Furthermore, we work under the assumption of a well mixed population, where any two people are equally likely to come into contact. This is also known as the *mass action* assumption, where the contact rate is independent of the population size. The alternative, *pseudo-mass action*, would assume the contact rate grows along with increasing population size.

We thus have a dynamical system governed by the following pair of ordinary differential equations (ODEs):

$$\frac{dS}{dt} = -\frac{\beta}{N} SI, \tag{2.1}$$

$$\frac{dI}{dt} = \frac{\beta}{N} SI. \tag{2.2}$$

These describe the change in susceptibles and infectious classes over time. The term $\beta SI/N$ is known as the transmission term, or *infection rate*: the rate at which new cases are generated in the population. It is the product between the force of infection and the number of susceptibles S. In our notation, we write β/N explicitly to reflect the mass-action assumption.

2.2.1 Demographic stochasticity

The set of ODEs above describe a *deterministic* system. If we were to simulate several epidemics with this system, given a fixed set of parameters we would obtain the same exact epidemic each time: every person would become infected at the same time across simulations. Naturally, this does not correspond to the dynamics of real-world epidemics, where randomness plays an important role. *Stochastic* models attempt to approximate this behaviour.

A popular stochastic model, known as *demographic stochasticity*, uses an event-based approach. This is most commonly implemented using the Gillespie algorithm, which uses the cumulative rates of possible events (e.g. infection or recovery) to determine the time to the next event, and what the next event will actually be [Gillespie, 1977].

Event	Transition	Rate
Infection	$S \to S - 1$ $I \to I + 1$	$\beta SI/N$

Table 2.1: Definition of the SI stochastic system.

Table 2.1 above summarises the stochastic SI system. As we can see, the only possible event is an infection. This increases the number in the infectives pool by 1, and decreases the number in the susceptible pool by the same number. Therefore, the only stochastic element in an SI system is the time between successive events: the *interevent* times (also commonly called *waiting* times). If we assume these interevent times to be approximately exponentially distributed (discussed further in Chapter 4), we can draw the time to next event δt randomly from an exponential distribution with mean $(\beta SI/N)^{-1}$. For an SI epidemic, we iterate this process until no more susceptibles remain.

2.2.2 Behaviour and visualisation

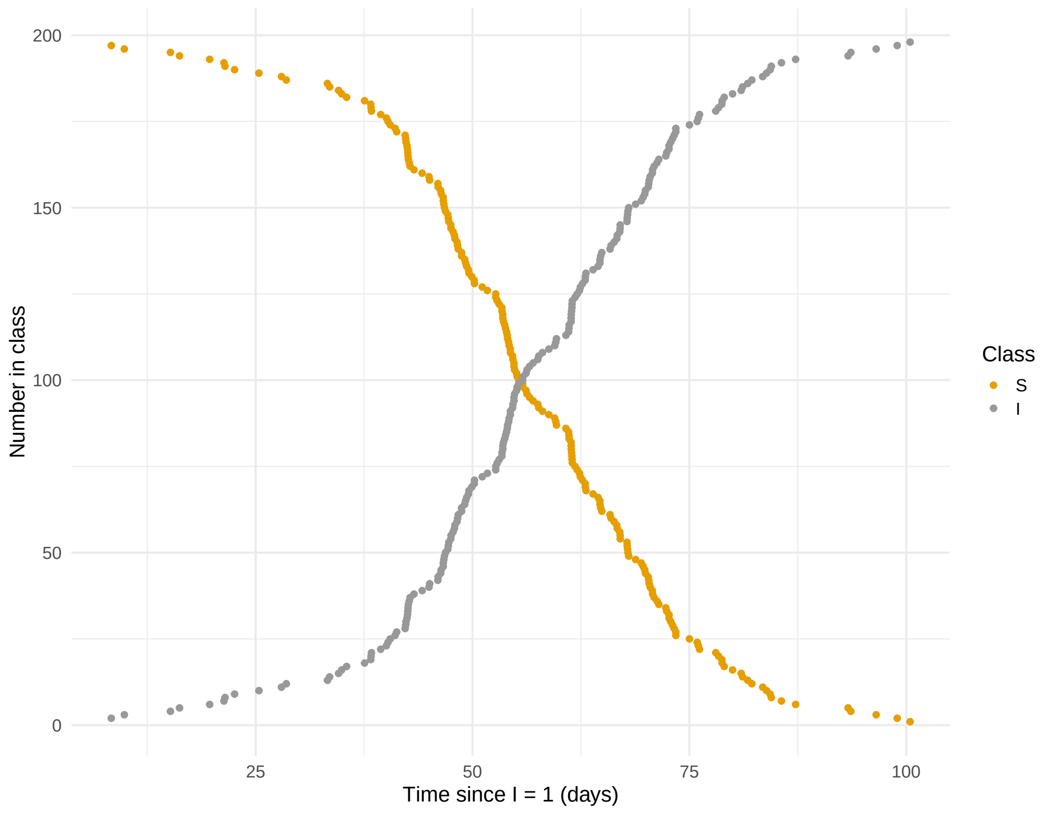

Figure 2.2: SI model behaviour, simulated stochastically with $\beta = 0.1$ and $N = 200$.

A useful property of the SI model is that it can be rewritten using $S = N - I$. Explicitly, $\beta SI/N = \beta(N-I)I/N$. The equation describes logistic growth: initially the number of infectives grows exponentially, and slows down as the susceptible pool is depleted.

Figure 2.2 above visualises the dynamics in a simple SI epidemic. The epidemic was simulated using the stochastic method described previously. Solely looking at class I, each point in the figure corresponds to an *infection event*. Therefore, the time between them are the interevent times.

2.3 Without immunity: the SIS model

The basic SI model is generally an oversimplification of the transmission dynamics of many infectious diseases: there is no immunity, and all susceptibles eventually become infectious. For sexually transmitted diseases or rotaviruses for example, there is some immunity, but it is not permanent. An individual may become contract the agent multiple times during a lifetime. In terms of compartmental models, we refer to this as a recovery followed by a direct return to the susceptible pool. In other words, as soon as an individual is no longer infectious, they become immediately susceptible again.

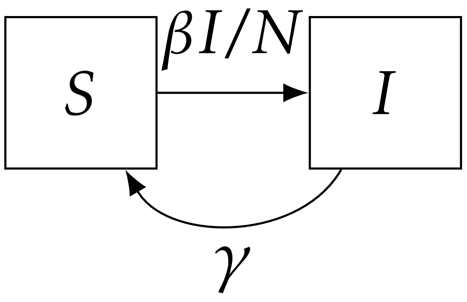

Figure 2.3: Susceptible-infectious-susceptible (SIS) model.

We can model these type of diseases using a susceptible-infectious-susceptible (SIS) model, as shown in Figure 2.3 above. Visually, it is similar to the basic SI model, with the exception of a new parameter γ: the rate of recovery. Generally, we are more interested in its reciprocal $1/\gamma$, which is the average infectious period. As before, the population is closed with $S+I=N$. The ODEs governing this model are

$$\frac{dS}{dt} = -\frac{\beta}{N}SI + \gamma I, \tag{2.3}$$

$$\frac{dI}{dt} = \frac{\beta}{N}SI - \gamma I. \tag{2.4}$$

The assumption of a constant (i.e. independent of time of infection) recovery rate γ has important biological implications. Specifically, it implies the duration of infectivity τ is exponentially distributed with mean $1/\gamma$. Therefore, those who have been infectious for longer periods of time contribute more to transmission of the agent than those who have been infectious for only a short amount of time.

2.3.1 Stochastic formulation

In an SIS epidemic, two possible events may occur: an infection, or a recovery. Therefore, in addition to determining the time to the next event, we must also determine *which* event must occur. Table 2.2 below summarises the SIS stochastic system.

Event	Transition	Rate
Infection	$S \to S-1$ $I \to I+1$	$\beta SI/N$
Recovery	$S \to S+1$ $I \to I-1$	γI

Table 2.2: Definition of the SIS stochastic system.

If an infection occurs at a rate of $\beta SI/N$, and a recovery occurs with γI, either event must occur with a rate of $\beta SI/N + \gamma I$. The reciprocal of the sum of rates is thus the mean of the exponential distribution from which we will draw the time to the next event. To determine which event occurs next, we use the ratio of recovery to total rate $\gamma I/(\beta SI/N + \gamma I)$, and draw a random uniform deviate U. If the ratio is larger than U, the next event is a recovery, and vice-versa.

2.3.2 Behaviour and visualisation

With this added recovery parameter, we must also introduce a critical epidemiological quantity: $R_0 = \beta/\gamma$, the *basic reproductive ratio*. This is defined as the average number of secondary cases an infectious individual produces in a fully susceptible population. An infection can only invade if $R_0 > 1$, or it will exponentially decay. Intuitively, this makes sense: the infection parameter β must be larger than the recovery rate γ in order for the disease to spread.

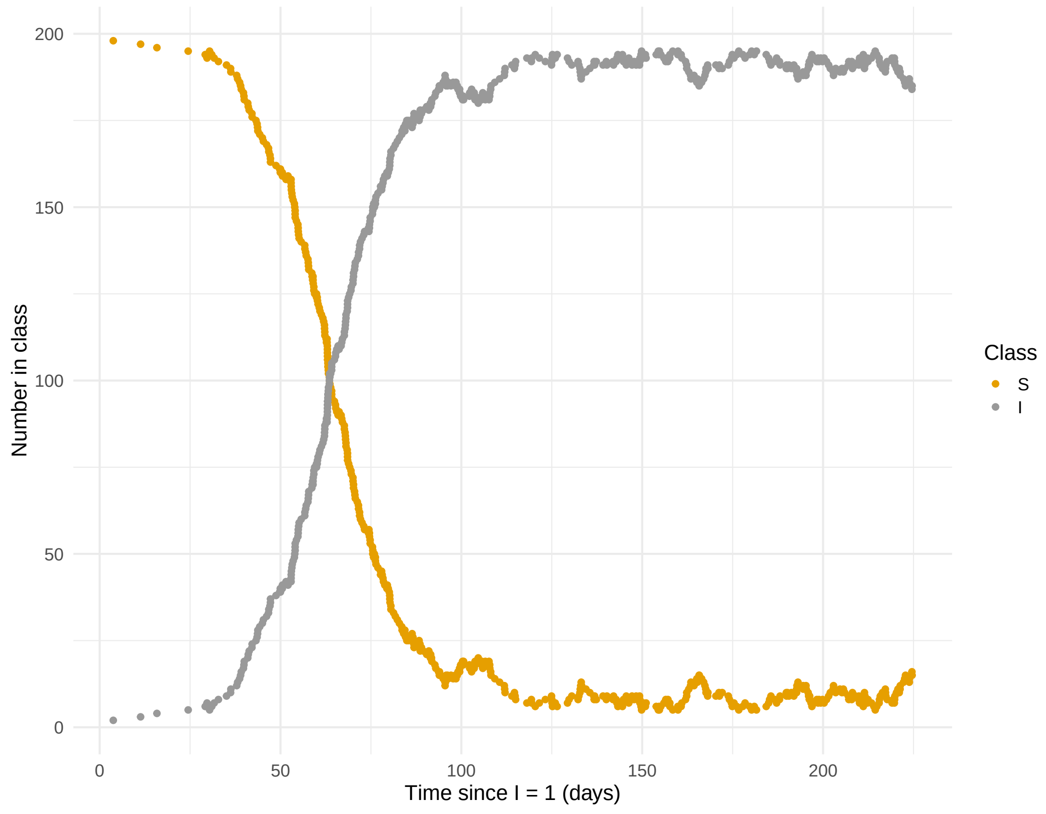

Figure 2.4: SIS model behaviour, simulated stochastically with $\beta = 0.1$, $N = 200$ and $\gamma = 1/200$ (average infectious period of 200 days).

Figure 2.4 above gives insight into the behaviour of SIS-type epidemics. Looking solely at class I again, we can see that after around the 100 day mark, I stays approximately constant: the epidemic has reached an *endemic equilibrium*. For an SIS epidemic, this is only possible when $R_0 > 1$ and the equilibrium is defined as $I^* = N(1-1/R_0)$. As this is computed by solving $dI/dt = 0$, the only other existing equilibrium is $I^* = 0$. This is the disease-free state, which is stable when $R_0 < 1$.

2.4 Lasting immunity: the SIR model

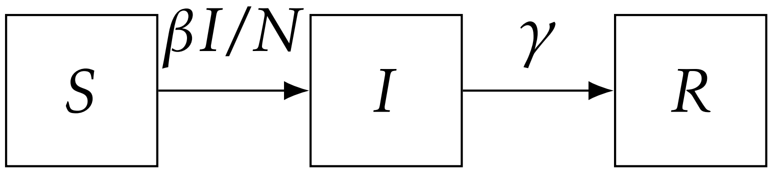

Figure 2.5: Susceptible-Infectious-Recovered (SIR) model.

Also known as the general epidemic model, the SIR model in Figure 2.5 extends the SIS model by adding an additional class: R, or *recovered* [Kermack and McKendrick, 1927]. In other words, patients who are no longer infectious move to the R compartment where they have acquired indefinite immunity to the disease. Like in the SIS context, the rate of recovery is again constant, and also denoted γ. The corresponding ODEs are:

$$\frac{dS}{dt} = -\frac{\beta}{N}SI, \tag{2.5}$$
$$\frac{dI}{dt} = \frac{\beta}{N}SI - \gamma I, \tag{2.6}$$
$$\frac{dR}{dt} = \gamma I. \tag{2.7}$$

2.4.1 Stochastic formulation

Event	Transition	Rate
Infection	$S \rightarrow S-1$ $I \rightarrow I+1$	$\beta SI/N$
Recovery	$R \rightarrow R+1$ $I \rightarrow I-1$	γI

Table 2.3: Definition of the SIR stochastic system.

Table 2.3 shows the SIR stochastic model. Like in the SIS case, only two events are possible. The process of generating the time to the next event and the nature of the next event is analogous to the SIS case. The only difference is a matter of bookkeeping: the number in either S or R will stay the same from one event to the next, depending on which of the two occurs. For example, in the case of an infection, we have $S \to S - 1,\ I \to I + 1$ and $R \to R$.

2.4.2 Behaviour and visualisation

Without demographics, there are only two equilibrium points for an SIR epidemic: an entirely susceptible or entirely recovered population. In the latter case, there are no births to refill the susceptible pool: it will eventually become depleted. Figure 2.6 below shows the progression of a typical SIR epidemic. Here we clearly see that the epidemic does not end when the susceptible pool is depleted, but rather when the pool of infectives is depleted, and everyone has subsequently recovered.

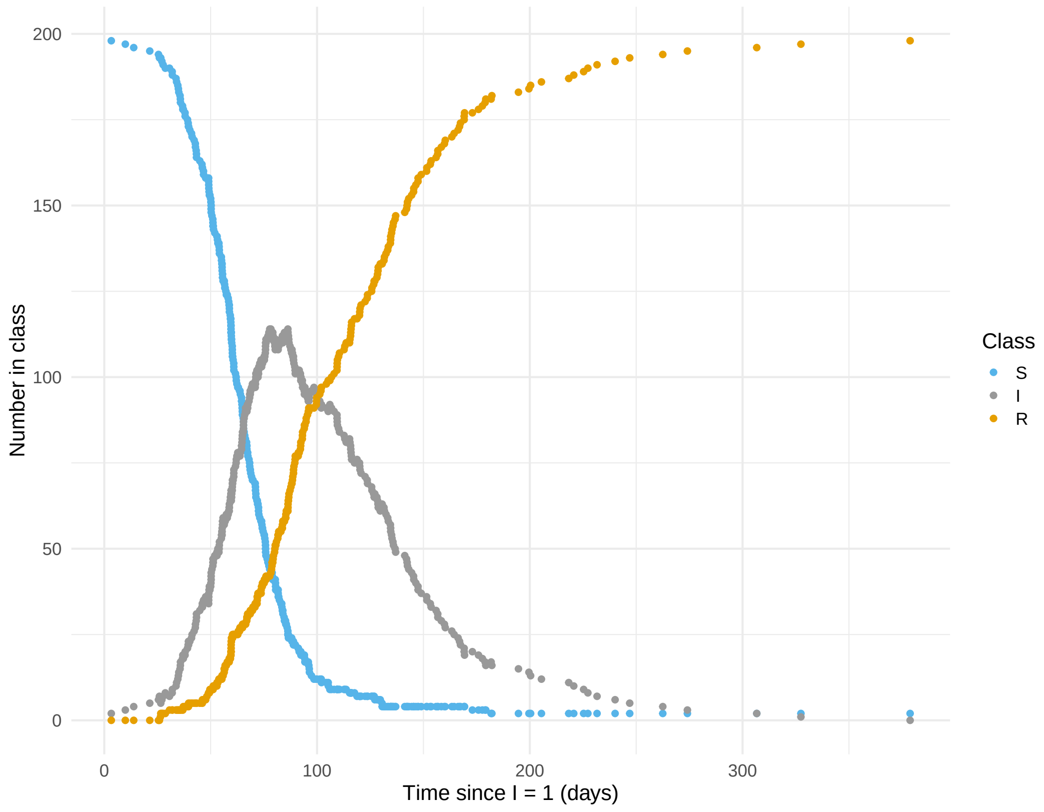

Figure 2.6: SIR model behaviour, simulated stochastically with $\beta = 0.1$, $N = 200$ and $\gamma = 1/50$ (average infectious period of 50 days).

Chapter 3

Gamma Generalised Linear Models

This chapter provides an introduction to Gamma generalised linear models (GLMs). We will go through the frequentist and Bayesian alternatives, with a focus on how to deal with the inverse link function. The main references for this chapter are McCullagh and Nelder [1989] and Faraway [2006] for the frequentist approach, and Gelman et al. [2004] for the Bayesian perspective, unless otherwise specified.

3.1 Recapitulation of GLMs and introduction of notation

3.1.1 GLM theory

A GLM extends the traditional linear regression model to situations with non-gaussian response variables, and where the relationship between predictors and response is not linear. Let $\mathbf{y}$, the response or *dependent* variable, be an $n \times 1$ vector of independently distributed realisations of a random variable Y with means $\boldsymbol{\mu}$. In the case of simple linear regression, we have independent Gaussian distributions with $\mathrm{E}(Y) = \boldsymbol{\mu}$ and constant variance σ^2. Specifying the probability distribution of Y is referred to as the *random component* of a GLM, and the distribution can be any in the exponential family of the form

$$f_Y(y,\theta,\phi) = \exp\left\{\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)\right\}, \tag{3.1}$$

where θ is the *canonical* parameter, ϕ is the *dispersion* parameter; $b(\cdot)$ and $c(\cdot)$ are some specific functions. It also follows that $\mathrm{E}(Y) = \mu = b'(\theta)$, and $\mathrm{Var}(Y) = b''(\theta)a(\phi)$.

Let $\mathbf{X}$ be an $n \times p$ matrix of explanatory variables $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_p}$. The *systematic component* of a GLM specifies the *linear predictor* $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$: the linear combination between the predictors and the $p \times 1$ vector of unknown regression coefficients $\boldsymbol{\beta}$.

The *link function* $\eta = g(\mu)$ describes the relationship between systematic and random parts of the model. The link $g(\cdot)$ must be monotonic and differentiable. In the case of simple linear regression, $\eta = \mu$, which corresponds to the identity link. The *canonical link* corresponds to $\eta = \theta$, where the link functions connects the expected value to the canonical parameter in the exponential family form. This link is mathematically convenient, partly because it simplifies the derivation of the maximum likelihood estimator (MLE).

The vector of coefficients $\boldsymbol{\beta}$ is estimated via maximum likelihood maximisation. Expressed in terms of expected value μ, we denote the log-likelihood of a single data point as $\ell(\mu, \phi | y) = \log f_Y(y, \mu, \phi)$. For a data vector $\mathbf{y}$, we sum up the individual contributions of all data points, yielding

$$\ell(\boldsymbol{\mu}, \phi | \mathbf{y}) = \sum_{i}^{n} \log f_Y(y_i, \mu_i, \phi). \tag{3.2}$$

3.1.2 Gamma GLM

For data where the response variable is positively skewed, continuous and strictly non-negative, Gamma GLMs are useful (also known in literature as *Gamma regression*, see Wasef Hattab [2016]). They are a special case of GLMs where we assume the dependent variable y follows a gamma distribution, with density

$$f(y,\alpha,\beta) = \frac{\beta^{\alpha} y^{\alpha-1} e^{-\beta y}}{\Gamma(\alpha)}; \text{ for } y > 0, \text{ and } \alpha, \beta > 0, \tag{3.3}$$

where α is the *shape* parameter, β is the *rate* parameter and Γ is the gamma function. Panel A in Figure 3.1 plots gamma distributions with varying shape, and rate parameter held constant. Panel B again plots different gamma distributions, this time varying the rate and holding the shape constant at $\alpha = 1$: this corresponds to the exponential distribution. All densities in the figure are right-skewed.

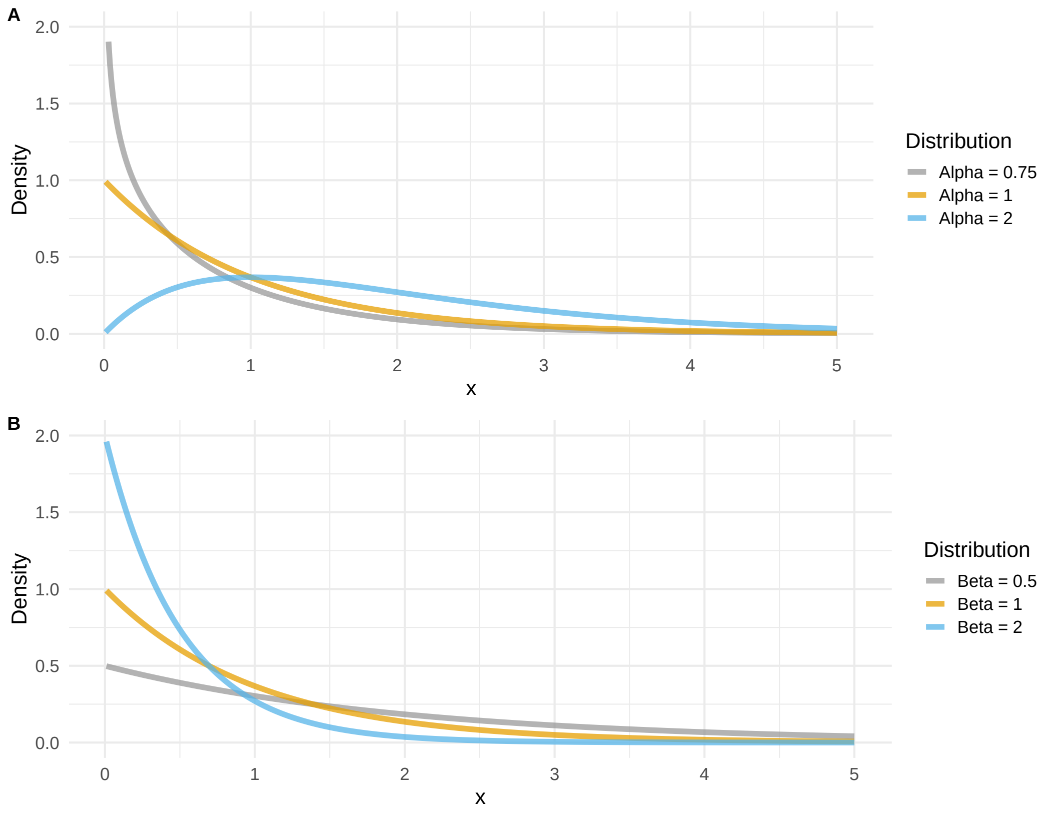

Figure 3.1: Panel A plots different shapes α for constant rate of $\beta = 1$, and panel B varies the rate while the shape is held constant at $\alpha = 1$ (exponential).

In the context of GLMs, it is convenient to reparametrise the gamma density by setting $\mu = \alpha/\beta$. This implies $\mathrm{E}(Y) = \mu$ and $\mathrm{Var}(Y) = \mu^2/\alpha$: the variance is proportional to the mean squared. Furthermore, the dispersion parameter is defined as $\phi = \alpha^{-1}$. For brevity, we can write $Y \sim \Gamma(\mu, \mu^2/\alpha) \equiv \mathrm{Gamma}(\mu, \mu^2/\alpha)$. The log-likelihood contribution of a single data point is

$$\ell(\alpha, \mu | y) = \alpha \log \frac{\alpha}{\mu} - \log \Gamma(\alpha) + (\alpha - 1) \log y - \frac{\alpha}{\mu} y. \tag{3.4}$$

3.1.3 Goodness of fit

In the setting of GLMs, the *deviance* is often used as a measure of goodness of fit. It measures how close the current model is to a perfect fit: when $\hat{\boldsymbol{\mu}} = \mathbf{y}$. For a Gamma GLM, the deviance is defined as

$$D(\mathbf{y}, \hat{\boldsymbol{\mu}}) = -2 \sum \left\{ \log \frac{y_i}{\hat{\mu}_i} - \frac{y_i - \hat{\mu}_i}{\hat{\mu}_i} \right\}. \tag{3.5}$$

This quantity is proportional to the twice the difference between $\ell(\alpha, \hat{\boldsymbol{\mu}} = \mathbf{y} | \mathbf{y})$ and $\ell(\alpha, \hat{\boldsymbol{\mu}} | \mathbf{y})$. That is to say, between the estimated log-likelihood versus the maximum attainable log-likelihood. The `glm()` function in `R` provides both the *residual* and *null* deviance. Consider the *saturated model*, which has as many parameters as it has data points: it represents a perfect model fit (maximum possible log-likelihood). The residual deviance represents the discrepancy the saturated model and the proposed model. The null deviance on the other hand compares the saturated model to an intercept-only model (null model).

3.1.4 Link function

In addition to the identity link ($\eta = \mu$), there are two other common choices of $g(\cdot)$ for Gamma GLMs. The first is the log link, $\eta = \log \mu$, which may be useful if the predictors have a multiplicative effect on the mean. Furthermore, this link function assures positive predicted values $\hat{\mu}$, which is an attractive property since $\eta \in \mathbb{R}^+$.

The second function is the inverse link, $\eta = \mu^{-1}$, which is the canonical link. This link is of particular interest in this thesis, since the estimates have a convenient rate interpretation. Contrastingly to the log link, the inverse link does not ensure $\hat{\mu} > 0$. Hence, the $\eta > 0$ condition implies restrictions must be taken when estimating $\hat{\boldsymbol{\beta}}$. Furthermore, Gamma GLMs with an inverse link often fail to converge if appropriate starting values are not provided. Most research exploring Gamma GLMs acknowledge restrictions must be taken, but fail to suggest any practical solutions [Winkelmann, 2008, Zuur et al., 2013]. From Section 3.2 to the end of this chapter, we propose a Bayesian approach to dealing with the inverse link. Chapter 4 then proceeds to showing how we can use Gamma GLMs with an inverse link to model infection rates.

3.2 Solving issues with the inverse link: a Bayesian approach

3.2.1 Philosophy

Let $\boldsymbol{\theta} = \{\alpha, \boldsymbol{\beta}\}$ summarise the parameters in a Gamma GLM. In *frequentist* statistics (like above), we assume that the true parameters $\boldsymbol{\theta}^*$ are fixed, but unknown. It assumes that given enough samples, the parameter estimates will eventually converge to their true values.

In *Bayesian* statistics, we take $\boldsymbol{\theta}^*$ to be a random variable and condition on observed data $\boldsymbol{y}$. Given this data, a Bayesian updates priors beliefs about θ into a posterior belief, for which we use Bayes' rule

$$p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{3.6}$$

where $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ is the *likelihood function* of the data, $p(\boldsymbol{\theta})$ is the *prior* distribution of the parameters and $p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X})$ is the *posterior* distribution. On the log-scale, this posterior distribution is the sum of the data log-likelihood, and the log-density of the prior.

3.2.2 Likelihood function

For a Gamma GLM with inverse link, the log-likelihood function is simply the gamma log-likelihood specified in (3.4). For a particular data set, we can write this as: $\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \ell(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X})$. As the sample size n increases, this function has an increasingly large influence on the posterior distribution. Of note however, is that in the case of the inverse link this function is only defined for $0 < \mu$. Thus, we can more specifically write the log-likelihood function of the model as follows:

$$\ell(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \begin{cases} -\infty & \text{if } \mu \leq 0, \\ \sum_{i=1}^{n} \ell(\alpha, \mu_i|y_i) & \text{Otherwise.} \end{cases} \tag{3.7}$$

3.2.3 Choice of priors

For a Gamma GLM, we can rewrite the prior as $p(\boldsymbol{\theta}) = p(\alpha)p(\boldsymbol{\beta}|\alpha)$. That is to say, we need to set a prior distribution for the shape parameter α, and another for the vector of coefficients β conditional on the shape. Concerning the regression coefficients $\boldsymbol{\beta}$, Gelman et al. [2008] suggest the use of Cauchy distributions as weakly informative priors, rather than uninformative flat priors. Specifically, for an intercept term we may use a Cauchy density with center 0 and scale 10, and for other covariates we can adjust the scale parameter to 2.5.

Any log-concave prior would be appropriate for the shape $\alpha > 0$. Gelman [2006] recommends the use of the half-Cauchy distribution (the right, positive side of the symmetric Cauchy distribution) as a weakly informative prior, while Elfadaly and Garthwaite [2015] use a more informative lognormal prior.

3.2.4 Sampling from the posterior

In practice, it is difficult to obtain an analytical solution for the posterior density. Instead, we can sample from it even though we do not have $p(\boldsymbol{y})$ (the normalising constant) from expression (3.6). The sampling is done by a random walk over the parameter pace, following rules for the moves such that the marginal distribution of a stationary walk should correspond to the posterior distribution. We refer to this type of sampling as *Markov chain Monte Carlo* (MCMC), which brings together two important ideas.

The first is Monte Carlo simulation, which attempts to approximate the value of a parameter or quantity (e.g. expectation) by repeatedly drawing random numbers.

Second, a Markov chain is simply a sequence of random elements, in which the the next *state* in the chain depends solely on the current state, and not all past states. Formally, let $X^{(0)}, X^{(1)}, \dots$ be a sequence of random variables. This sequence is Markovian if $P(X^{(i+1)} = x^{(i+1)} | X^{(i)} = x^{(i)}, X^{(i-1)} = x^{(i-1)}, X^{(0)} = x^{(0)}) = P(X^{(i+1)} = x^{(i+1)} | X^{(i)} = x^{(i)})$.

3.2.5 Off-the-shelf MCMC samplers

For a Bayesian Gamma GLM with inverse link, there is little readily available software. One of them is Stan, and its applied regression modelling package 'rstanarm' [Carpenter et al., 2017, Gabry, 2018]. Variants of Hamiltonian MCMC are used, which are able to produce chains with very low autocorrelation between samples. To avoid negative values of μ when the inverse link is used, the sampler adjusts the intercept term to effectively shift η so as to make all values positive. This solution may introduce a possible bias in the samples. To see some example code, refer to appendix B.7.1.

Another popular MCMC sampler used for Bayesian GLMs is JAGS, which implements Gibbs sampling [Plummer, 2003]. Dealing with the inverse link in JAGS can be done in two ways. One can either use a similar intercept-driven strategy as described above to shift linear predictor values, or very informative (positive) priors can be used for the intercept and regression parameters. Particularly with the latter option, there is still no guarantee of avoiding log-likelihood computation errors: a single sample from the extremes of the posterior could still yield negative linear predictor values, which would lead to an undefined log-likelihood.

3.2.6 A tailored MCMC sampler for a Gamma GLM

One possible MCMC sampler we could use is the *Metropolis-Hastings* (MH) algorithm [Hastings, 1970]. The success of a MH sampler rests on two main components. First, the *acceptance probability* $a(\cdot)$ ensures the sampler visits areas of higher density more often, and avoids getting stuck locally by being able to restore its previous move in the parameter space.

Second, the *proposal function* $q(\cdot)$ is critical, as it drives the exploration of the parameter space. To illustrate, let us assume we want to sample from a normal distribution with $\mu = 2$ and $\sigma = 5$ using MH. We pick a normal distribution again with mean 2 as our proposal function. If we were to pick a very small σ for our proposal, the sampler would have a very high acceptance rate (ratio of accepted proposals to total proposals), but would be very slow to explore the entire target distribution. If σ was too large however, the acceptance rate would be low as the sampler may explore areas that are far removed from the target. An optimal $q(\cdot)$ thus allows an efficient exploration of the parameter space, by balancing the ratio of accepted to rejected proposals.

For a Gamma GLM, we need to choose a proposal distribution for the shape parameter and regression coefficients. For $0 < \alpha$ to hold, a possible proposal distribution for the shape is the lognormal, centred at the log of the previous value. The standard deviation on the log-scale is slightly more arbitrary. If we assume the dependent variable to be approximately exponential, we may set this standard deviation to 0.1, implying proposals that will rarely leave the range $\alpha \in [0.5, 1.5]$.

To construct the proposal distribution for $\boldsymbol{\beta}$, we can make use of the fact that in a Gamma GLM, the variance is proportional to the mean squared. We can mimic this behaviour in a weighted simple linear model: $1/y = \eta$. Using a diagonal matrix of weights $\mathbf{W} = \text{diag}(y^2)$, we then minimise the weighted sums of squares:

$$WSS = \underset{\beta}{\text{argmin}} \|\mathbf{W}^{1/2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\|^2. \tag{3.8}$$

From this model extract the estimated variance-covariance matrix of the parameters $\hat{\Sigma}$, and use it as part of a multivariate normal proposal. To optimise further, we can scale this matrix by a factor of $2.38^2/d$, as per Rosenthal and Others [2008]. Here d corresponds to the dimension of $\boldsymbol{\beta}$, and we denote the scaled matrix Σ.

We can then rejection sample this proposal density such that proposed values of $\boldsymbol{\beta}$ that do not satisfy the $0 < \mu$ (and thus $0 < \eta$) restriction are not evaluated in the acceptance function. This is in fact a case *advanced rejection* of values, rather than sampling from a truncated distribution per se. Instead of rejection sampling, we could equivalently set the acceptance probability to zero as soon as one of the restrictions are violated. Our proposal distribution here is thus the product of a lognormal, and a multivariate normal distribution. Algorithm 1 summarises the MH for the Gamma GLM with inverse link, including the rejection sampling step.

Algorithm 1: Truncated random-walk Metropolis-Hastings for Gamma GLM with inverse link.

```
1  From weighted linear model, extract $\hat{\boldsymbol{\beta}}$ ($d$-dimensional) and $\hat{\Sigma}$ ;
2  Initialize parameters $x^{(0)}$: $\boldsymbol{\beta}^{(0)} = \hat{\boldsymbol{\beta}}$, $\alpha^{(0)} = 1$ and $\Sigma = \hat{\Sigma} \times 2.38^2/d$ ;
3  **for** *iterations* $i = 1, 2, \ldots, m$ **do**
4      Propose $\alpha' \sim \text{LogNorm}(\log(\alpha^{(i-1)}), 0.1)$ ;
5      **repeat**
6          Propose $\boldsymbol{\beta}' \sim \mathcal{N}(\boldsymbol{\beta}^{(i-1)}, \Sigma)$ ;
7      **until** $0 < \eta$;
8      Candidate state $x' = (\boldsymbol{\beta}', \alpha')$ has been drawn according to proposal
         distribution $q(x'|x^{(i-1)})$ ;
9      Acceptance probability $a(x'|x^{(i-1)}) = \min\left\{1, \frac{p(x')q(x^{(i-1)}|x')}{p(x)q(x'|x^{(i-1)})}\right\}$, where $p(\cdot)$
         is the posterior density function ;
10     Draw $u \sim \mathcal{U}(0, 1)$ ;
11     **if** $u < a$ **then**
12         Accept proposal, set $x^{(i)} \leftarrow x'$ ;
13     **else**
14         Reject proposal, set $x^{(i)} \leftarrow x^{(i-1)}$ ;
15     **end**
16     Set $i = i + 1$ ;
17 **end**
```

A note must also be made on the symmetry of the proposal distribution. The acceptance probability introduced in line 9 of Algorithm 1 can be simplified to a simple ratio of posterior densities, *if* the proposal distribution is symmetric. However, the current proposal function is composed of a symmetric multivariate normal and an asymmetric lognormal distribution. Correcting for the lognormal is trivial, and requires the simple use of the lognormal density. Correcting for the asymmetry is important because failing to do so means we will be sampling from a posterior that will most likely be very close to the target distribution, but not exactly the target.

Finally, a possible development regarding the introduced MH sampler is to make it *adaptive*. Specifically, we can monitor the acceptance rate during a burn-in phase, and adaptively scale the variance-covariance matrix Σ of the multivariate normal proposal accordingly. By doing so, we can tune the final acceptance rate to a desired value. The scaling is done during burn-in to conserve the ergodic properties of the Markov chain. Nevertheless, adaptation during burn-in is not necessarily preferred relative to non-adaptive MH: if adaptation is performed before proper convergence of the chains, we would be essentially adapting to a distribution that is not the target distribution. To see corresponding code and further description, see appendix B.4.2. For the remainder of the thesis, we proceed with the non-adaptive MH.

3.2.7 Outcomes

From the posterior distribution of the parameters, several quantities are of interest. In addition to common quantities of the distribution like the mean, median and various quantiles; the *maximum a posteriori* (MAP) estimate is also important. This is the mode of the posterior distribution, the one that maximises its density. One of the ways we can find this MAP estimate is by using *simulated annealing* [Kirkpatrick et al., 1983, Johansen et al., 2007].

For a certain function $f(x)$, the idea is to progressively concentrate the probability mass around the mode(s). This can be done by using $f(x) \propto (f(x))^{\beta}$, where $\beta \to \infty$. In the current MH, this would be implemented at the acceptance probability stage. Specifically, we would raise the ratio of posterior densities to the power of β, and increase β steadily across iterations. This can be done using *geometric tempering*. Let $\beta^{(0)} = 1$ be the first value of β. Then at each iteration i, we implement $\beta^{(i)} = \alpha^{t} \times \beta^{(i-1)}$, where $\alpha^{t} > 1$ is the tempering constant. Figure 3.2 gives an example of what this annealing might look like in practice. The mode of $f(x)$ is then estimated as the mode of the annealed distribution.

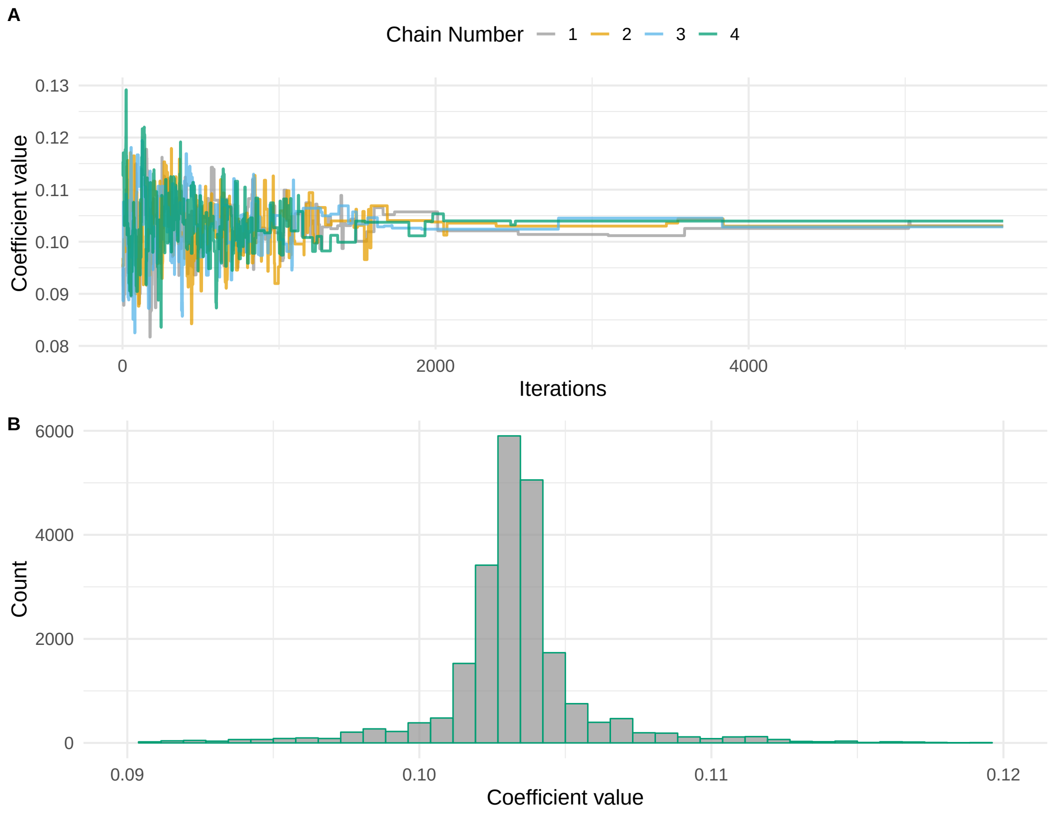

Figure 3.2: Example of simulated annealing with geometric tempering ($\alpha^t = 1.002$).

3.2.8 Diagnostics

Once the MAP estimate and associated quantiles are computed, it is also essential to assess the quality of the Markov chain. In addition to the general acceptance rate, traceplots (such as panel A from Figure 3.2) give insight into the mixing and convergence properties of the chains. In other words, if we run multiple parallel Markov chains we can see whether are they all exploring the desired parameter space, or if they are diverging and getting stuck in local areas (which may be dependent on starting values). Furthermore, the Gelman-Rubin diagnostic $\hat{R}$ can also be used to assess convergence [Gelman and Rubin, 1992]. For each parameter, the difference in variance between and within chains is assessed: values of $\hat{R}$ close to 1 (and smaller than 1.1) are an indicator of suitable convergence. The Gelman-Rubin diagnostic is readily available using the R package ‘coda’ [Plummer et al., 2006].

A second property to assess is *autocorrelation*; the extent to which samples in a chain are correlated. Ideally, we would like this to be as small (independent) as possible, to ensure we have an accurate picture of our target distribution. This can be assessed using autocorrelation plots, which plot the lag-k correlation between states $x^{(i)}$ and $x^{(i+k)}$ of the chain.

Chapter 4

Rethinking epidemic models using interevent times

The present chapter brings together the previous two chapters on infectious disease modelling and Gamma GLMs respectively. Specifically, we look at how a Bayesian Gamma GLM can be used to model the infection interevent times in SI (Section 4.2), SIR (Section 4.3) and SIS-type (Section 4.4) epidemics.

4.1 Observable events and interevent times

When forecasting infectious diseases, we are mainly concerned with the dynamics of observable infection events over time. It would thus be useful to gauge what the infection rate is at different points in time or as a function of observable quantities, such as cumulative counts of infection (C), or prevalence of infectives in the population (I). Infection interevent times may provide a possible way to do this.

Let t_i denote the time of i^{th} infection, $i \in \{0, 1, ..., n\}$ with n being the total number of infection events. We refer to the interevent times as $y_i = t_i - t_{i-1}$, for $i \in \{1, 2, ..., n\}$. Of interest here would be the infection rate, which is the reciprocal of the expected interevent time. Explicitly, if infection events occur on average every μ days, then we can say new cases are generated at a rate of $1/\mu$.

To get an insight into these interevent times, we can take the infectives from the previously simulated epidemics and plot the corresponding interevent times against the number of infectives, as seen in Figure 4.1 below.

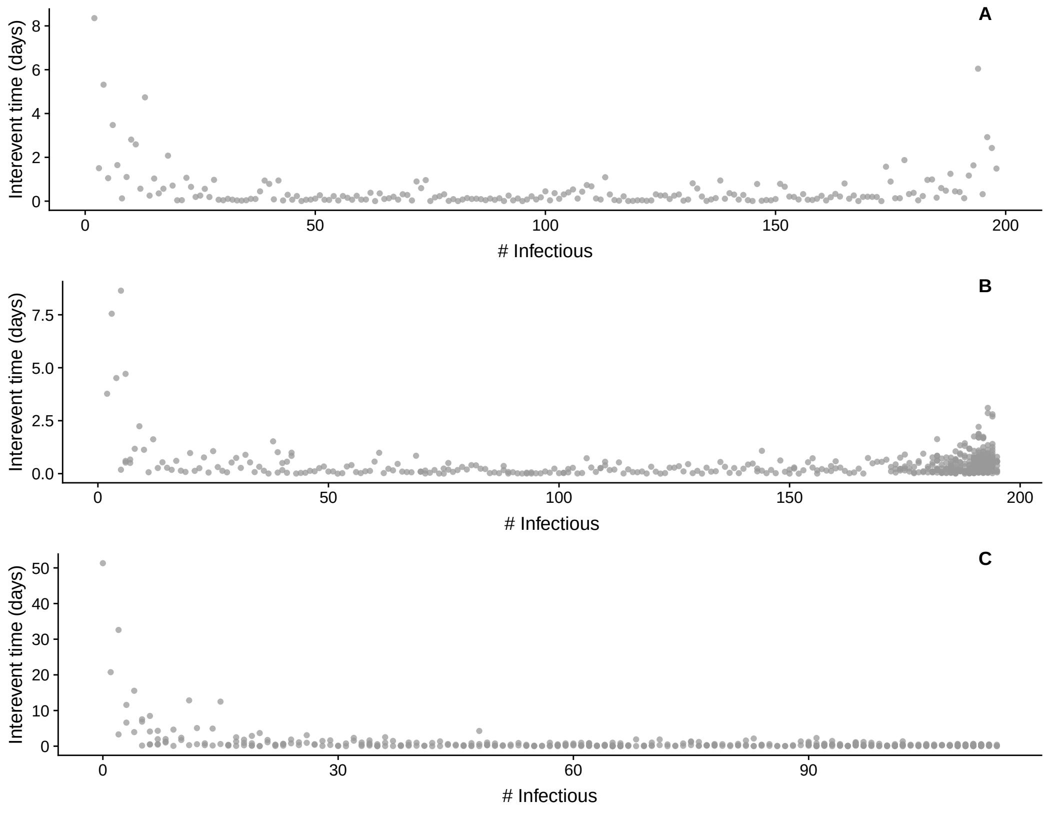

Figure 4.1: Infection interevent times plotted against I, using infectives from figures 2.2 (panel A, SI-type), 2.4 (panel B, SIS-type) and 2.6 (panel C, SIR-type).

In this figure, we observe some interesting behaviour. First, for all three models the interevent times are much larger when the number of infectives is low (at the beginning of the epidemic). For the SI model (panel A), the interevent times also become larger towards the end, leading to a characteristic 'bucket shape'. In the SIS case, the equilibrium is very visible from accumulation of points on the right of the interevent plot (panel B).

In panel C (SIR), the interevent times don't accumulate or become larger where I is higher. The plot doesn't immediately reflect the infection interevent times being larger at the beginning and end of the epidemic. If we imagine the points being plot sequentially, the plot would be filled from left to right first, and then back again in an almost mirrored fashion.

4.2 A model for SI-type epidemics

4.2.1 Observed events formulation

So far, we have been working with I, the number of infectives. In the event of a real epidemic, it is near impossible to keep track of I. Instead, we can work with the number of notified/reported infections C and try to model the reciprocal of the reported interevent times dC/dt. C is also referred to as the event counter, or the cumulative number of events.

For SI dynamics, we can reformulate as follows. We observe the number of reported infection events C with under-reporting rate $1/\rho$. Thus, ρC are the actual number of infection events. As an example, if $1/\rho = 0.5$, then only observe half of the actual number of events are observed. Using $S + I = N$,

$$\begin{aligned}\frac{dC}{dt} &= \frac{1}{\rho}\frac{\beta}{N}SI,\\ &= \frac{1}{\rho}\frac{\beta}{N}(N-I)I,\\ &= \frac{1}{\rho}(\beta I - \frac{\beta}{N}I^2).\end{aligned} \tag{4.1}$$

Let S_t, I_t and C_t denote the number of susceptibles, infectives and reported infection events at time t respectively. It follows that $S_t = S_0 - \rho C_t,\ \ I_t = I_0 + \rho C_t$ and $C_0 = 0$. By substitution,

$$\begin{aligned}\frac{dC}{dt} &= \frac{1}{\rho}\frac{\beta}{N}(S_0 - \rho C_t)(I_0 + \rho C_t),\\ &= \frac{1}{\rho}\frac{\beta}{N}S_0 I_0 + \frac{\beta}{N}(S_0 - I_0)C_t - \rho\frac{\beta}{N}C_t^2.\end{aligned} \tag{4.2}$$

Equation (4.2) thus expresses the reciprocal of the observed actual interevent rate in terms of the actual number of observed infections for SI-type epidemics.

4.2.2 GLM formulation, and interpretation of regression coefficients

Let μ be the expected reported interevent time, and x be an explanatory variable containing the cumulative number of reported infections ($x = C$). We can write the GLM model with an inverse link as:

$$\frac{1}{\mu} = \eta = \beta_0 + \beta_1 x + \beta_2 x^2, \tag{4.3}$$

with gamma distributed errors, mean μ and shape parameter α. Taking (4.2), the data-generating model for reported infections is:

$$\frac{1}{\mu} = \eta = \frac{1}{\rho}\frac{\beta}{N} S_0 I_0 + \frac{\beta}{N}(S_0 - I_0)C_t - \rho\frac{\beta}{N}C_t^2, \tag{4.4}$$

which implies,

$$\beta_0 \approx \frac{1}{\rho}\frac{\beta}{N} S_0 I_0, \tag{4.5}$$

$$\beta_1 \approx \frac{\beta}{N}(S_0 - I_0), \tag{4.6}$$

$$\beta_2 \approx -\rho\frac{\beta}{N}. \tag{4.7}$$

By rearranging the previous terms, we can also write $-\beta_1/\beta_2 \approx (S_0 - I_0)/\rho$, which is the total number of infections that will be reported. Under the data-generating model, we expect the errors to be gamma distributed with shape $\alpha > 1$. This stems from the assumption that the time between actual infection events I are approximately exponentially distributed: due to under-reporting, the interevent times for reported infections will be more sparse (i.e. the gamma distribution will be more skewed).

4.2.3 Sampler restrictions

Following from the interpretation of coefficients above, the $0 < \eta$ restriction discussed in the previous chapter must become more specific. The total number of expected reported infections $(S_0 - I_0)/\rho$ must be strictly positive, and should be retrievable from the (negative) ratio between a *positive* β_1 and a *negative* β_2. We can summarise the restrictions on the model parameters as:

$$0 < \alpha, \quad 0 < \eta, \quad 0 < \beta_1, \quad 0 > \beta_2. \tag{4.8}$$

To clarify, the $0 < \alpha$ restriction is imposed via the prior distribution. The latter three constraints do not stem from the prior, but rather from the definition of the model log-likelihood in (3.7). If any one of the above restrictions are violated, the log-likelihood is simply $-\infty$ and we must continue to rejection sample the proposal distribution in the MH sampler.

4.3 A model for SIR-type epidemics

4.3.1 Observed events formulation

For SIR dynamics, let us express I_t as function of S_t:

$$\frac{dI}{dS} = \frac{\frac{\beta}{N}SI - \gamma I}{-\frac{\beta}{N}SI}, \tag{4.9}$$

$$\Leftrightarrow dI = (-1 + \frac{\gamma N}{\beta S})dS, \tag{4.10}$$

$$\Leftrightarrow I_t - I_0 = -S_t + S_0 + \frac{\gamma N}{\beta}\ln(\frac{S_t}{S_0}). \tag{4.11}$$

Substituting I_t,

$$\begin{aligned}\frac{dC}{dt} &= \frac{1}{\rho}\frac{\beta}{N}SI, \\ &= \frac{1}{\rho}\frac{\beta}{N}S_t(I_0 - S_t + S_0 + \frac{\gamma N}{\beta}\ln(\frac{S_t}{S_0}).\end{aligned} \tag{4.12}$$

Substituting $S_t = S_0 - \rho C_t$,

$$\frac{dC}{dt} = \frac{1}{\rho}\frac{\beta}{N}(S_0 - \rho C_t)(I_0 + \rho C_t + \frac{\gamma N}{\beta}\ln(1 - \frac{\rho C_t}{S_0})). \tag{4.13}$$

We can then use a first order Taylor approximation of $\ln(1-\rho C_t/S_0) \approx \rho C_t/S_0$, which works well if $\rho C_t/S_0$ is small. That is, if the epidemic affects a small proportion of all susceptibles, which is plausible for infections that are poorly transmitted. Using the approximation in (4.13) above yields:

$$\begin{aligned}\frac{dC}{dt} &\approx \frac{1}{\rho}\frac{\beta}{N}(S_0 - \rho C_t)(I_0 + \rho C_t + \frac{\gamma N}{\beta}\frac{\rho C_t}{S_0}),\\ &\approx \frac{1}{\rho}\frac{\beta}{N}S_0 I_0 + \frac{\beta}{N}(S_0 - \frac{\gamma N}{\beta} - I_0)C_t - \rho\frac{\beta}{N}(1 - \frac{\gamma N}{\beta S_0})C_t^2. \end{aligned} \tag{4.14}$$

This expresses the reciprocal of the observed actual interevent rate as an approximation in terms of the actual number of observed infections for SIR-type epidemics.

4.3.2 GLM formulation, and interpretation of regression coefficients

The recovered class in SIR-type epidemics means there is no easy way to express the model solely in terms of I, as is the case in SI or SIS-type epidemics. Nevertheless, we can use expression (4.14) to build a GLM model.

As in subsection 4.2.2, we take μ to be the expected interevent time, and $x = C$. The model again is of a quadratic form with an inverse link:

$$\frac{1}{\mu} = \eta = \beta_0 + \beta_1 x + \beta_2 x^2, \tag{4.15}$$

where observations follow a gamma distribution with mean μ and shape parameter α. The data-generating model for reported infections is approximated by:

$$\frac{1}{\mu} = \eta \approx \frac{1}{\rho}\frac{\beta}{N}S_0 I_0 + \frac{\beta}{N}(S_0 - \frac{\gamma N}{\beta} - I_0)C_t - \rho\frac{\beta}{N}(1 - \frac{\gamma N}{\beta S_0})C_t^2, \tag{4.16}$$

the observations approximately following a gamma distribution with $\alpha > 1$, with the same reasoning as the SI case. Thus,

$$\beta_0 \approx \frac{1}{\rho}\frac{\beta}{N}S_0 I_0, \tag{4.17}$$

$$\beta_1 \approx \frac{\beta}{N}(S_0 - \frac{\gamma N}{\beta} - I_0), \tag{4.18}$$

$$\beta_2 \approx -\rho\frac{\beta}{N}(1 - \frac{\gamma N}{\beta S_0}), \tag{4.19}$$

$$x \approx C_t, \tag{4.20}$$

which implies the in-sampler parameter restrictions are the same as those outlined in subsection 4.2.3. The only difference lies in how we retrieve the eventual total number of reported infections. We can calculate it by taking the largest root of $\beta_0 + \beta_1 x + \beta_2 x^2 = 0$, since we cannot express it directly as a function of β_1 and β_2.

4.4 A model for SIS-type epidemics

4.4.1 Observed events formulation

For SIS dynamics, we can write analogously to the SI case (4.1):

$$\begin{aligned}\frac{dC}{dt} &= \frac{1}{\rho}\frac{\beta}{N}SI,\\ &= \frac{1}{\rho}(\beta I - \frac{\beta}{N}I^2).\end{aligned} \tag{4.21}$$

Let Q_t be the number of 'observed' recovery events, where we only count the recovery of people who were observed to be infected. We then have $I_t = I_0 + \rho C_t - \rho Q_t$, with $C_0 = 0$, $Q_0 = 0$. By substitution,

$$\frac{dC}{dt} = \frac{1}{\rho}(\beta(I_0 + \rho C_t - \rho Q_t) - \frac{\beta}{N}(I_0 + \rho C_t - \rho Q_t)^2). \tag{4.22}$$

Furthermore, due to a constant recovery rate, the duration of infectivity τ is distributed as

$$\tau \sim \gamma e^{-\gamma \tau}, \tag{4.23}$$

with mean $1/\gamma$. It follows that $Q_t = I_0 \gamma e^{-\gamma t} + \int_0^t \gamma e^{-\gamma t} C(t-\tau) d\tau$, is an exponentially smoothed version of C_t. We can approximate $Q_t \approx I_0 \gamma e^{-\gamma t} + C(t - 1/\gamma)$ using Jensen's inequality, since C_t is convex:

$$C(t-1/\gamma) = C(\int_0^t \gamma e^{-\gamma t}(t-\tau)d\tau) \leq \int_0^t \gamma e^{-\gamma t} C(t-\tau) d\tau. \tag{4.24}$$

Thus, this approximation is an underestimation of Q. If I_0 is negligible, we have:

$$\frac{dC}{dt} \approx \beta(C_t - C_{t-1/\gamma}) - \rho \frac{\beta}{N}(C_t - C_{t-1/\gamma})^2, \tag{4.25}$$

or alternatively,

$$\frac{dC}{dt} \approx \beta C_t - \rho \frac{\beta}{N} C_t^2 \underbrace{- \beta C_{t-1/\gamma} - \rho \frac{\beta}{N} C_{t-1/\gamma}^2 + 2\rho \frac{\beta}{N} C_t C_{t-1/\gamma}}_{\text{Delay terms}}, \tag{4.26}$$

where the delay terms are characteristic for SIS when compared to SI, SIR dynamics. We have thus expressed the reciprocal of the observed actual interevent rate as an approximation in terms of the actual and past number of observed infections for SIS-type epidemics.

4.4.2 A risk-set approximation of I

The delay terms in (4.26) make a GLM model as function of the cumulative number of past infections rather difficult. Instead, it would be convenient to create an explanatory variable x that approximates I, and takes into account that infected individuals stay infectious only for a period of γ^{-1} on average. This would enable us to the data-generating model in (4.21).

To do so, we take a risk-set approach, which is analogous to what is done in left-truncated survival analysis. Let $t_{rec} = t_i + \gamma^{-1}$. In other words, t_{rec} is the sum of the time since first infection, and the average duration of infectivity. We assume this t_{rec} to be the event of interest (i.e. a recovery), and that all patients experience the event, while 'entering the study' (becoming infected) at time t_i. Doing so, we can keep track of the number of people at any given time in this hypothetical study, and use it as an approximation of I.

t	$t + \gamma^{-1}$	Approximate I (x)
5	340	1
345	680	1
360	695	2
370	705	3

Table 4.1: Risk-set approach to constructing I. Here we assume t is time since first infection (days), and duration of infectivity $\gamma^{-1} = 335$ days (as an example, which means infectious individuals become susceptible again after approximately 11 months). Each row is a single patient.

Table 4.1 gives an example of what this approach looks like. In the beginning of an epidemic, where the interevent times tend to be rather large, this approach may yield one or more zero values in the constructed I. To avoid zero values in the eventual GLM analysis, we set any $x < 1$ to 1.

4.4.3 GLM formulation

With this newly created explanatory variable x, the model is again of a quadratic form with inverse link:

$$\frac{1}{\mu} = \eta = \beta_0 + \beta_1 x + \beta_2 x^2. \tag{4.27}$$

The data-generating model is:

$$\frac{1}{\mu} = \eta = \beta I - \frac{\beta}{N} I^2, \tag{4.28}$$

where the observations are approximately gamma distributed. Like in the SI case, we have $\beta_0 \approx 0$, $\beta_1 \approx \beta$, $\beta_2 \approx -\beta/N$ and $x \approx I$. Furthermore, $N = -\beta_1/\beta_2$. This means the parameter restrictions in the MH sampler are exactly the same as those outlined in subsection 4.2.3.

Chapter 5

Forecasting

The current chapter looks at how we can use the estimators in Chapter 4 to forecast future number of reported infections in all three epidemic types. We then discuss in Section 5.5 how we can evaluate the calibration of the obtained forecasts.

5.1 Bayesian forecasting

Predicting the number of reported infection events is performed in two main steps. Namely, running the Bayesian Gamma GLM on the infection interevent times, and then using the estimated parameters to stochastically simulate future infection events with variants of Gillespie's direct method (described in Chapter 2). This method effectively incorporates two different sources of forecast uncertainty.

First, the Bayesian Gamma GLM yields an entire posterior distribution of coefficients rather than just one set of maximum likelihood coefficient estimates (with associated confidence intervals), as would have been the case with a frequentist GLM. A single vector of samples $\boldsymbol{\theta}^{(i)} = [\alpha^{(i)}, \boldsymbol{\beta}^{(i)}]$ at iteration i of an MH chain contains enough information to make a single forecast. Therefore, a chain of hundreds or thousands of samples from the posterior distribution has enough information to make hundreds or thousands of forecasts. By definition, the discrepancies between all of these forecasts (even when made in a deterministic way) are partly due to uncertainty stemming from unknown parameter values.

Second, the vector $\boldsymbol{\theta}$ contains an estimate of α, the shape parameter in the gamma distribution of interevent times, and regression coefficients $\boldsymbol{\beta}$, which can be used to calculate the linear predictor η for each value of covariate. We can thus randomly draw the time for the next infection event using a Gamma distribution with shape α and rate $\eta \times \alpha$ (or equivalently α/μ). This type of stochastic model incorporates uncertainty because of random noise in a process that will accumulate or average out over the prediction interval. It is also important to note that using a Gamma GLM is advantageous here, as estimation of the shape parameter means we are not confined to strictly exponential interevent times, which may not always be the case with real-life epidemics.

5.2 Forecasting SI-type epidemics

To describe how forecasting is done for an SI-type epidemic, consider the data simulated in Figure 2.2. Specifically, let us only consider the data points before $I = 100$: half-way through the epidemic (panel A of Figure 5.1 below). The procedure is as follows:

1. Contruct y (infection intervent times), x (cumulative infections, in the current example $x = [1, 2, \ldots, 100]$) and x^2. Run the Bayesian Gamma GLM following Section 4.2.
2. Set a number T of desired forecasts smaller or equal to the total number of posterior samples. In case of the former, randomly select T posterior samples.
3. For each of these samples, set $\beta = \beta_1$, $N = -\beta_1/\beta_2$ (truncated downwards to the nearest integer), and record α.
4. Forecast stochastically from current cumulative number of infections $x_t = I$ (here $x_t = 100$) until $I = N$.

In step 4, we are effectively using the Gillespie algorithm to draw the time to the next infection event successively at each I. For an SI-type epidemic, as I

approaches N, the rate in the Gamma distribution of interevent times decreases, leading to larger interevent times being drawn. If we were to run the procedure on only three posterior samples, the result would be akin to that of panel B in Figure 5.1.

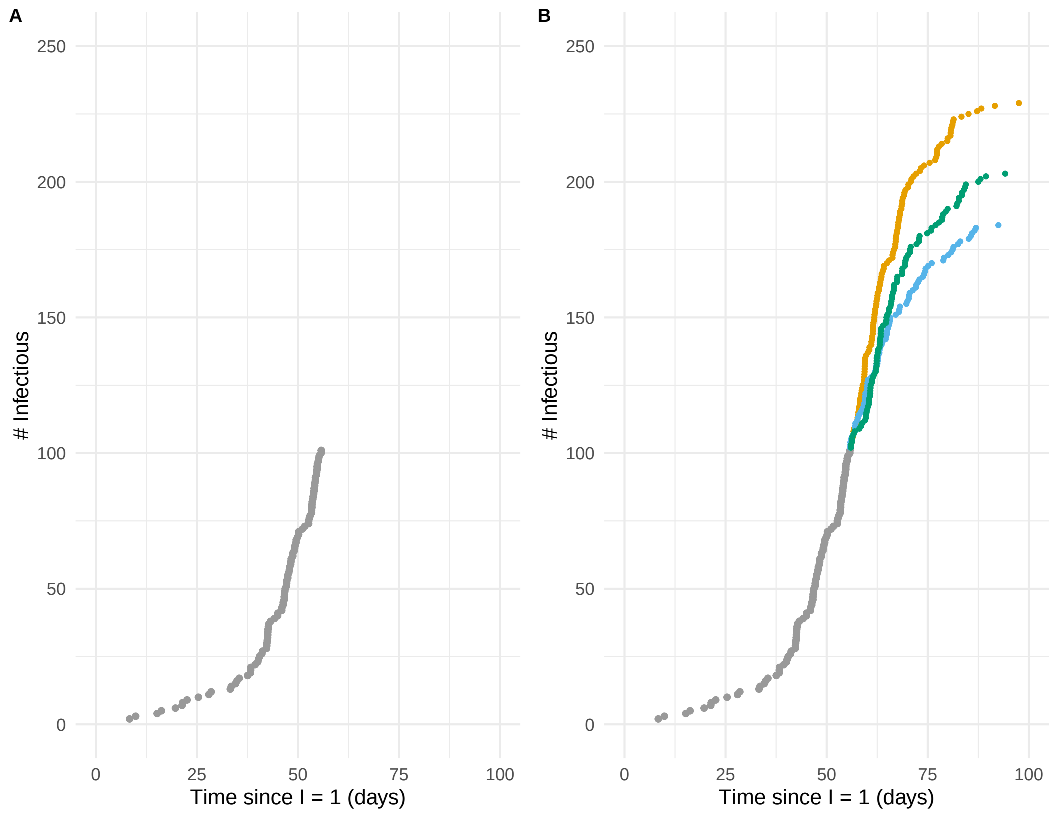

Figure 5.1: For a SI-type epidemic: example of applying a Bayesian Gamma GLM to the data in panel A, and using three draws from the posterior distribution to make three forecasts of future I (panel B).

Still considering panel B, suppose we are only interested in forecasting 20 days ahead. The value of I in all three forecasts at time $t = t + 20$ make up the *posterior predictive distribution* at that specific time point. The more posterior samples we use to forecast, the more detailed this distribution becomes. We can thus create *credible prediction intervals* at different points in time, to get a sense of whether the forecasts become more or less precise the further ahead we look. To do so, we look at the 5^{th} and 95^{th} percentiles over time.

5.3 Forecasting SIR-type epidemics

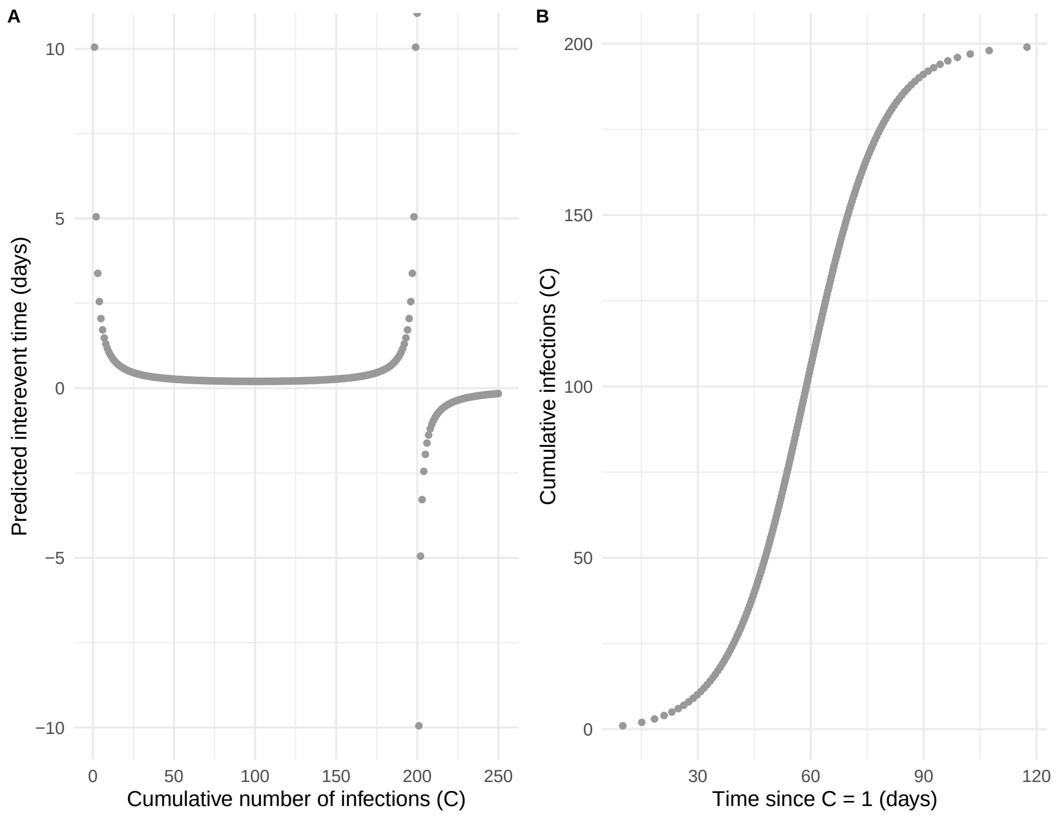

Figure 5.2: Panel A: predicted values of μ in a hypothetical Gamma GLM, with $\beta_0 = 0$, $\beta_1 = 0.1$ and $\beta_2 = -0.0005$. The behaviour around $C = 200$ (largest root of $\beta_0 + \beta_1 x + \beta_2 x^2 = 0$) is typical of an inverse polynomial function. Panel B: visualising the same data in terms of C. The epidemic ends when the $\mu < 0$.

For SIR-type epidemics, the estimated coefficients from the Gamma GLM map on to the approximation to the data-generating model. The forecasting process is as follows:

1. Perform steps 1 and 2 from the SI forecasting procedure. Set $x_t = C_t$, the latest cumulative number of infections.

2. For each posterior sampler, successively draw the time to next infection from

Gamma($\alpha, \alpha/\mu$), at each iteration increasing x_t by 1 and consequently re-computing μ.

3. Forecast ends at a pre-set forecasting horizon, or when the epidemic has died out. The latter occurs when $\eta \leq 0$.

To understand step 3, it is helpful to visualise the behaviour of inverse polynomials, as in panel A of Figure 5.2 above. Here, we deterministically predict the time to the next event using a hypothetical set of estimated coefficients $\boldsymbol{\beta} = [0, 0.1, -0.0005]$. The interevent time is thus computed as $\mu = (0 + 0.1 \times C - 0.0005 \times C^2)^{-1}$. At $C = 200$, $\eta \leq 0$ and so μ is undefined. For greater values of C, the predicted expected values are negative.

Stochastically, panel A would not be realisable as we cannot draw negative interevent times: the rate parameter of a gamma distribution must be strictly positive. Forecasting stochastically in the SIR-case must cease when $\eta \leq 0$, resulting in a curve similar to panel B.

5.4 Forecasting SIS-type epidemics

In the SIS case, we can execute a similar procedure to the SI case, with the added layer of also simulating recovery events. A priori, we fix γ such that γ^{-1} is the average duration of infectivity for a disease (as established from previous literature).

The Bayesian GLM is then run on the infection interevent times, using the risk-set approximation of I as the explanatory variable, as described in the previous chapter. Doing so allows us to use the posterior distribution to compute several quantities. Namely: the population size $N = -\beta_1/\beta_2$, the reproductive ratio $R_0 = \beta_1/\gamma$ and the endemic equilibrium $I^* = N(1-1/R_0)$. We can build credible intervals for all three of these quantities.

Concerning forecasting, we use the Gillespie algorithm as described in subsection 2.3.1. For each posterior sample, we set $\alpha = \hat{\alpha}/2$, where $\hat{\alpha}$ is the MAP estimate. This reflects the fact that two different events are possible, with a total rate of $\text{Rate}_{\text{Tot}} = \beta SI/N + \gamma I$. The time to next event (recovery or infection) is thus successively drawn from Gamma($\alpha, \text{Rate}_{\text{Tot}} \times \alpha$).

In the case of $1 < R_0$, SIS-type diseases converge towards a positive, non-zero equilibrium. This means that there is no ‘set time’ at which we must stop forecasting. Thus, it is preferable to iterate the interevent time draws until a forecasting horizon (e.g. two years ahead). After running the procedure on a multitude of posterior samples, the resulting predictive distribution will look similar to that of panel B in Figure 5.1.

5.5 Evaluating calibration: probability integral transform

In addition to summary statistics about the forecasts (median, 5^{th} and 95^{th} percentiles), it would be useful to evaluate how consistent they are. In a real epidemic, this is complicated to evaluate because at the time of making the forecast, we do not know what the true future incidence will be at different points in time. Instead, we evaluate the performance of the current forecasting procedure in a simulation experiment.

Let $F_t(x)$ be the predictive cumulative distribution function (CDF) at time t. Given a large enough number of forecasts T, we can approximate $F_t(x)$ using the *empirical* predictive CDF, of the form:

$$\hat{F}_t(x) = T^{-1} \sum_{t=1}^{T} \mathbf{1}\{x_t \leq x\}, \tag{5.1}$$

where $\mathbf{1}(\cdot)$ is an indicator function. Let x_t be the true value of I (or C, in the SI and SIR case) at time t. The *probability integral transform* (PIT) value is simply $\hat{F}_t(x_t)$, the value of the empirical predictive CDF at the realised value [Gneiting et al., 2003].

For a certain disease-type (e.g. SI), we can simulate many datasets using the same parameters. For each of these, we keep all infection events until a chosen time t_{cutoff}, and forecast until time $t = t_{\text{cutoff}} + c$ (c defines how far the forecast horizon is). For each dataset we have the true simulated realisation x_t, so we can record the PIT value at time t.

After collecting all PIT values, one can visualise their density in a so-called PIT histogram [Carney and Cunningham, 2006]. If the histogram is uniform, the forecasts are well-calibrated. In other words, the true data-generating distribution (we abbreviate it as DGD, for the rest of the thesis) is compatible with the predictive distribution at a given time t . Departures from uniformity are generally indicative of a biased model. Specifically, if the histogram has more mass to the left, the forecasts are on average too high; and if the mass is mostly on the right, the forecasts are too low.

The PIT can also be inform whether or not forecasts are overdispersed or underdispersed. Histograms showing a $\cap$-shaped distribution are indicative of overdispersed forecasts, while $\cup$-shaped distributions reflect underdispersed forecasts [Vannitsem et al., 2018]. It is sometimes easier to think of these two shapes in terms of the extremes of the DGD: in the overdispersed case, the extremes are rarely covered by the predictive distribution, while in the underdispersed case, they are disproportionately covered (i.e. the predictive CDF at the true value is either close to 0 or 1 very often).

Chapter 6

Calibration

This chapter evaluates the calibration of the forecast technique for SI, SIS and SIR-type epidemics using a simulation experiment, resulting in multiple PIT histograms. Section 6.1 below describes the experiment holistically, while the remaining sections showcase the resulting PIT histograms per epidemic type.

6.1 Simulation experiment

Following Section 5.5, it was important to evaluate the calibration of the forecasting technique for SI, SIS and SIR-type dynamics. To do so, a simulation experiment was performed. For each epidemic type, 1000 datasets were simulated stochastically, based on a fixed set of parameters. For all epidemic types, we fixed the population size at $N = 250$ and assumed the interevent times to be exponential ($\alpha = 1$). Values of β (and γ, for SIS and SIR) varied between epidemic types, and are described in greater detail in the following subsections.

All datasets were then cut off at 3 time points, spaced by 3 months along the growth phase of the epidemic. These cutoff points were chosen visually based on the deterministic evolution of the epidemic, given a set of parameters. At each cutoff, a Bayesian Gamma GLM was run (on the infection events preceding the cutoff point), and 100 stochastic forecasts were subsequently performed. If a simulated dataset post-cutoff contained less that 25 infection events, it was not used: it was assumed to have too small a sample size.

We then recorded the value of the empirical (predictive) CDF at the true simulated value of C_t or I at 3 different forecast horizons: 1 month (30.5 days), 6 months (183 days) and 1 year (366 days) ahead. The density of these values were then visualised using PIT histograms. This procedure resulted in 3 (epidemic types) × 3 (cutoff points) × 3 (forecast horizons) = 27 PIT histograms, 9 per epidemic type. All histograms used 15 bins, which is in between the recommended 10-20 bins [Gneiting et al., 2003]. This enabled inspection of whether the forecasting method displayed any bias dependent on epidemic type, cutoff point or prediction horizon.

6.2 PIT for SI-type forecasts

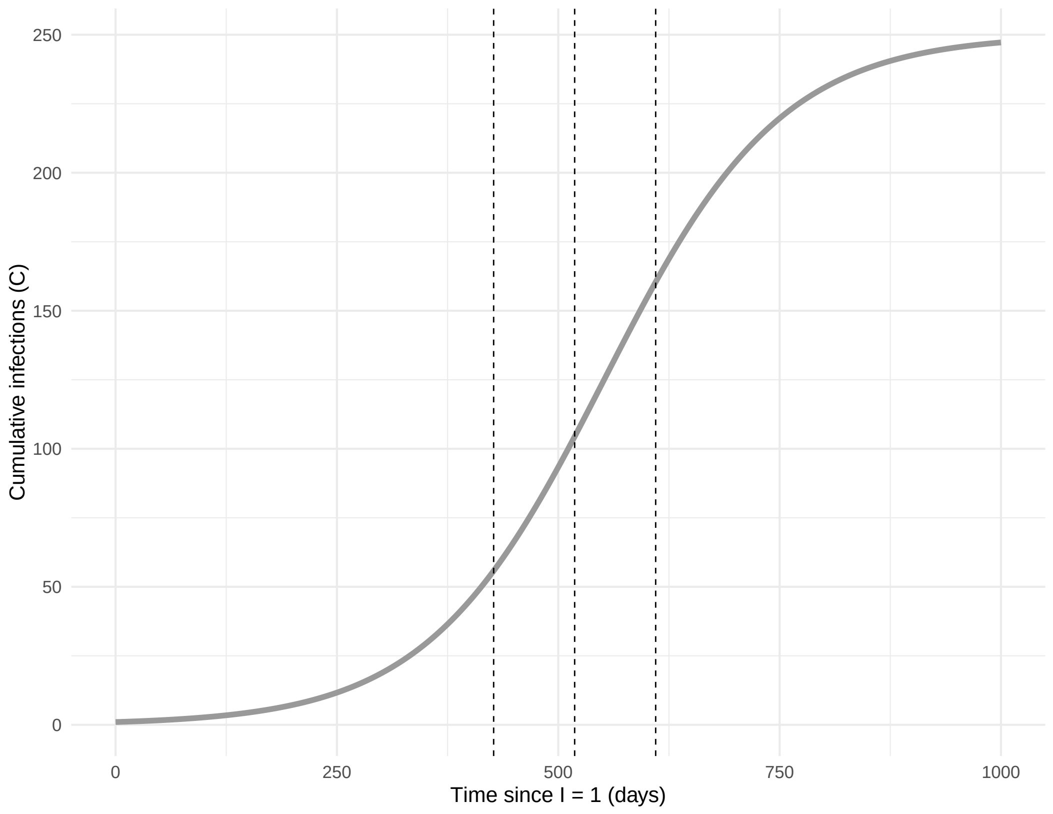

Figure 6.1: Deterministic course of cumulative infection counts for an SI-type epidemic with $\beta = 0.01$ and $N = 250$. The dashed lines are drawn at the chosen cutoff points $t_{\text{cutoff}} = [14, 17, 20]$ months after the first infection.

To evaluate the calibration of forecasts based on SI-type epidemic, $\beta = 0.01$ was fixed so as to have the total course of the epidemic last over 2 years. The cutoff points were chosen to be 14, 17, 20 months (427, 518.5, 610 days; assuming 1 month = 30.5 days), to cover the growth phase of the epidemic at intervals spaced by 3 months. This can visualised clearly in Figure 6.1 above.

Note that because we simulated *stochastically*, every dataset followed a slightly different time course. For example, at $t_{\text{cutoff}} = 14$ months, one dataset may have reached $C = 50$ cumulative counts (like in Figure 6.1), while another dataset may have already reached $C = 60$. Hence, by cutting-off the datasets *conditional* on a time rather than a C-based threshold (e.g. cut off all datasets at $t_{C=50}$), we ensured that we were not biased towards epidemics in an exponential growth phase.

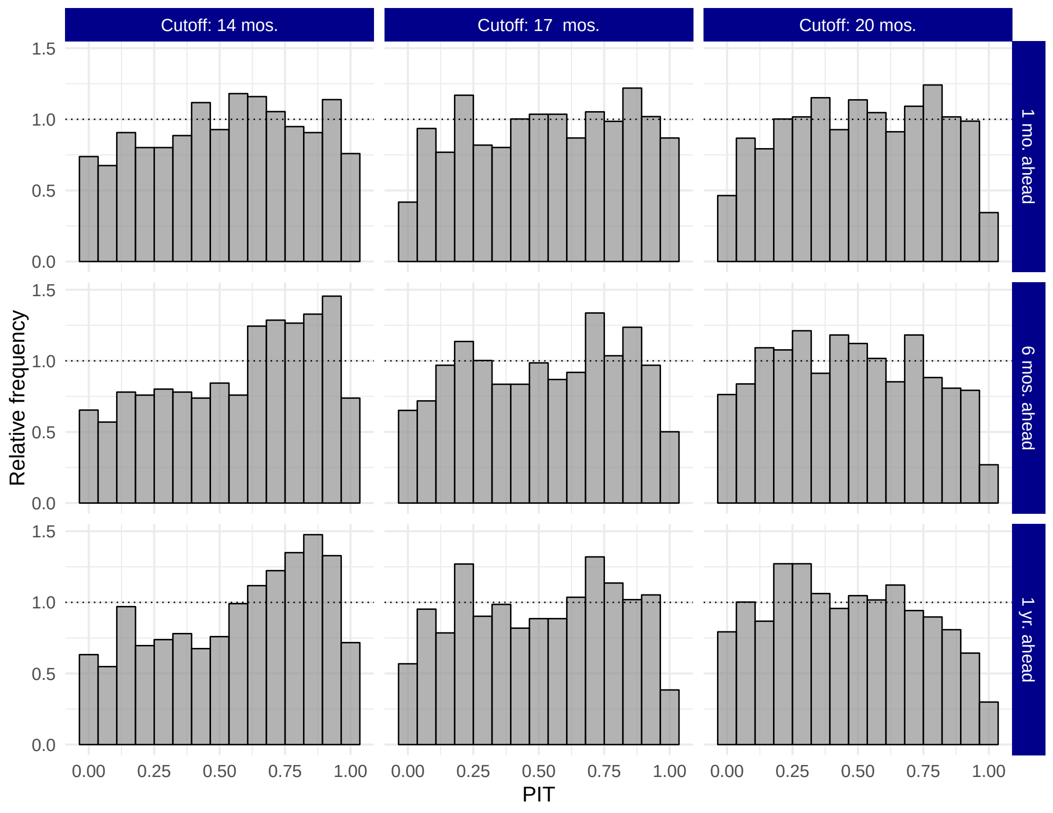

Figure 6.2: PIT histograms for SI-type simulated data cut off at $t_{\text{cutoff}} = [14, 17, 20]$ months; with 1 month, 6 months and 1 year ahead forecast horizons.

Figure 6.2 above shows the resulting PIT histograms. Due to only keeping datasets with $n > 25$, not all cutoff points had the same number of datasets to work with. For the cutoff at 14 months, 664 out of the 1000 simulated datasets had over 25 infection events. This was due to this time point being rather early in the (deterministic) growth phase of the epidemic. Cutoffs at 17 and 20 months in respectively were able to retain 838 and 936 datasets.

The PIT histograms are easiest to interpret column-wise, from first row to last: for each cutoff, and increasingly later horizons. If forecasts are perfectly calibrated, the histogram should be uniform (a dotted line is drawn to better detect departures from uniformity). A priori, we might expect forecasts at closer horizons to be calibrated better than those at later horizons. This is not unlike what we would expect for weather forecasting, where tomorrow's forecast is assumed to be more accurate than the forecast in one month's time.

For $t_{\text{cutoff}} = 14$ months, the 1 month ahead forecasts seemed relatively well calibrated, despite a small hint of overdispersion. In contrast, both later forecast horizons showed clear signs of miscalibration, with most mass placed on the right side of the histogram. This bias is reflective of chronic *underestimation*: the true values of C are more often found on the higher end of the predictive distribution, meaning most forecasts are below it. This bias seemed to be stronger in the 1 year ahead histogram compared to the 6 months ahead histogram.

At $t_{\text{cutoff}} = 17$, all three PIT histograms showed interesting behaviour: the extremes of the DGD were not particuarly well covered. In contrast, the area around the 25^{th} and 75^{th} percentiles seemed disproportionately well covered, which gave the histogram a slight hint of bimodality. The PIT values would have to be calculated for more simulated datasets before drawing any meaningful conclusions.

Finally, at $t_{\text{cutoff}} = 20$ months, with increasingly later forecasting horizons, the higher end of the DGD became less covered, which may be indicative of slight *overestimation.*

6.3 PIT for SIR-type forecasts

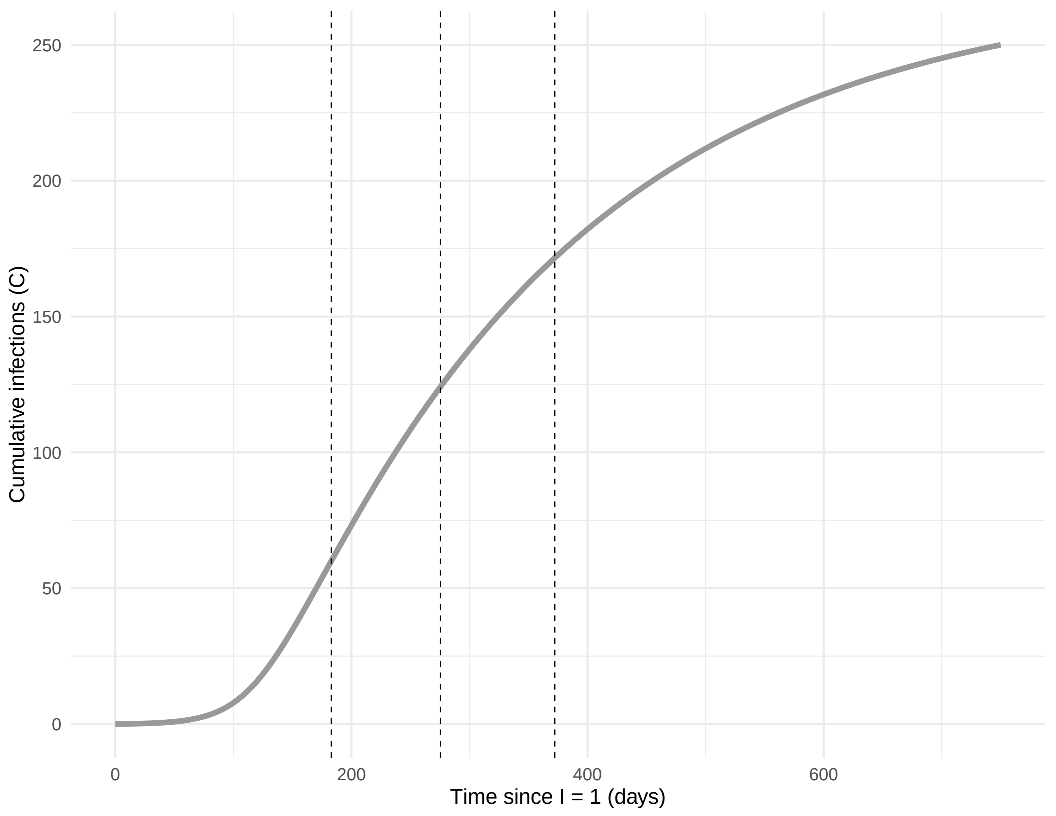

Figure 6.3: Deterministic course of cumulative infection counts for an SIR-type epidemic with $\beta = 0.05$, $\gamma^{-1} = 250$ days and $N = 250$. The dashed lines are drawn at the chosen cutoff points $t_{\text{cutoff}} = [6, 9, 12]$ months after the first infection.

For SIR-type epidemics, 1000 datasets were simulated using $\beta = 0.05$ and $\gamma^{-1} = 250$ days. Like in the SI case, the values were chosen to ensure the epidemic spanned at least 2 years. The cutoffs were chosen to be at 6, 9 and 12 months (183, 274.5, 366 days) in, which can be visualised in Figure 6.3 above. A small remark needs to be made regarding the shape of this cumulative counts curve, which doesn't possess the typical logistic shape (like in Figure 6.1). This was due to the recovery rate γ, which was comparatively much smaller (by a factor of 12.5) than the infection parameter β.

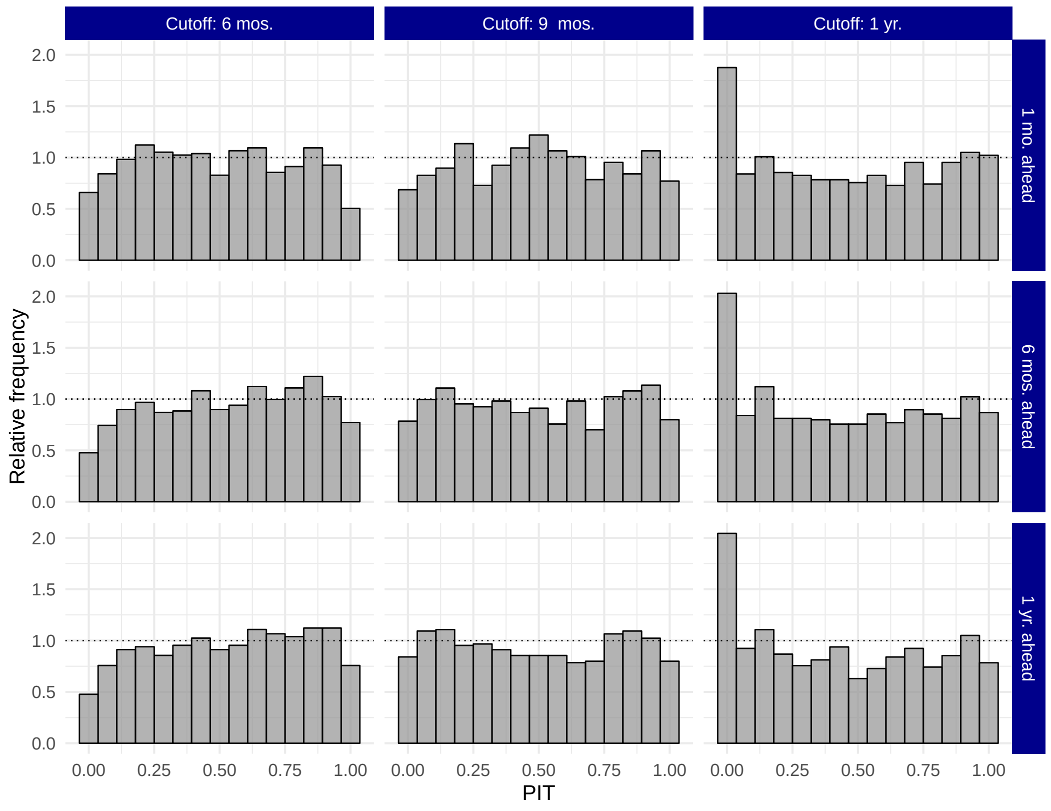

Figure 6.4: PIT histograms for SIR-type simulated data cut off at $t_{\text{cutoff}} = [6, 9, 12]$ months; with 1 month, 6 months and 1 year ahead forecast horizons.

Figure 6.4 shows the resulting PIT histograms after forecasting from all cutoff points. For cutoffs at 6, 9 and 12 months in; 998, 999 and all 1000 datasets respectively remained after checking for a minimum of 25 infection events. The PIT histograms for the 6 and 9 month cutoffs (for all forecast horizons) looked rather uniform and thus well calibrated. For the 6 month cutoff (6 month and 1 year ahead forecasts), the extremes of the DGD were not well represented. Importantly the departures from uniformity (when they occured) were not as extreme as in the PIT histograms for SI-type epidemics. This could be attributed to the comparatively higher number of simulated datasets with at least 25 infection events.

For $t_{\text{cutoff}} = 12$ months, the lowest 5% of the DGD was drastically overrepresented at all three forecast horizons, which was a clear indication of some overestimation. It was however unclear why neighbouring (higher) percentiles did not have as much mass in the histogram, and outside of the bottom 5% of the distribution, the rest of the histogram was rather uniform.

6.4 PIT for SIS-type forecasts

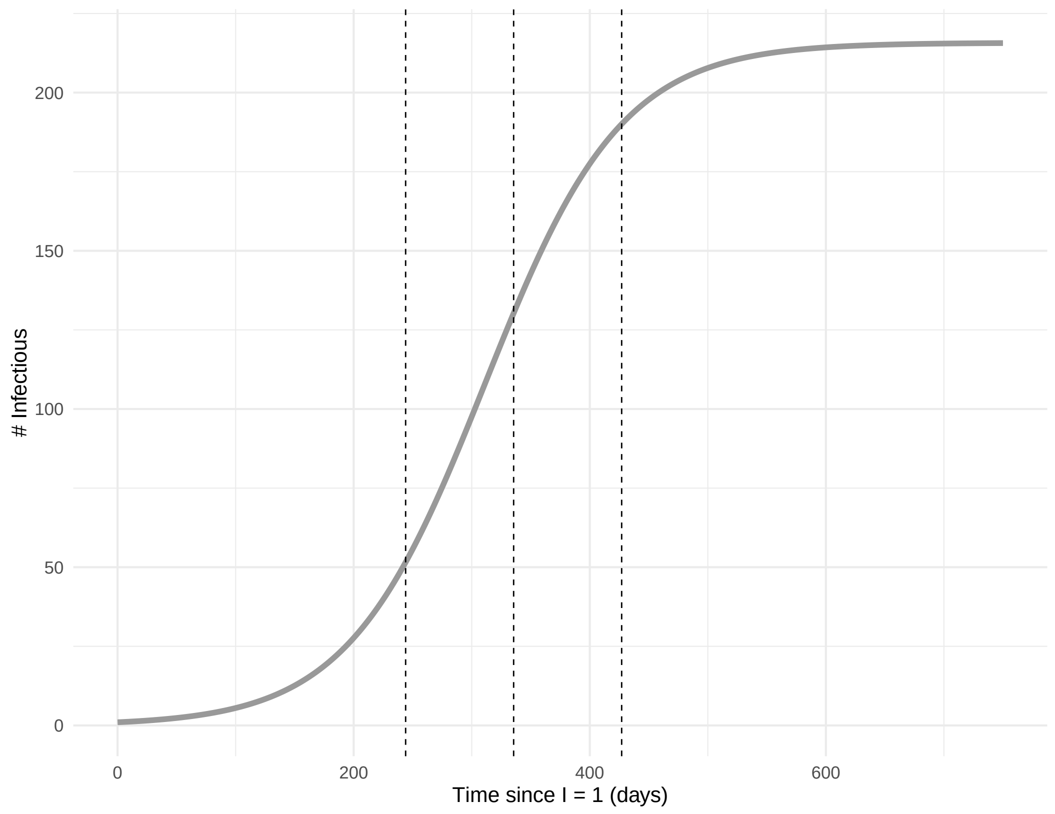

Figure 6.5: Deterministic course of I for an SIS-type epidemic with $\beta = 0.02$, $\gamma^{-1} = 365$ days and $N = 250$. The dashed lines are drawn at the chosen cutoff points $t_{\text{cutoff}} = [8, 11, 14]$ months after the first infection.

The parameters chosen to simulate 1000 SIS-type epidemics were $\beta = 0.02$ and $\gamma^{-1} = 365$ days. Cutoffs were chosen as 8, 11, 14 months (244, 335.5, 427 days) since the first infection event, as seen in Figure 6.5 above.

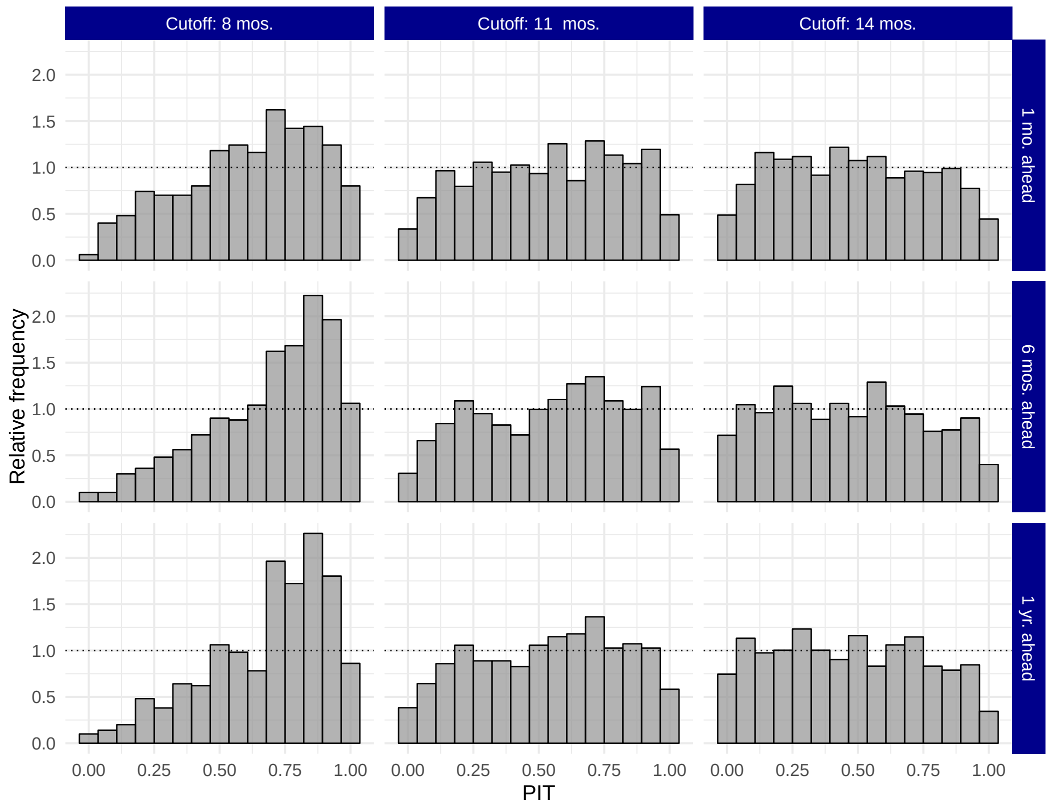

Figure 6.6: PIT histograms for SIS-type simulated data cut off at $t_{\text{cutoff}} = [8, 11, 14]$ months; with 1 month, 6 months and 1 year ahead forecast horizons.

Figure 6.4 above shows the resulting PIT histograms. By keeping datasets with minimum 25 infection infection events; 699, 914, 977 datasets remained for cutoff points at 8, 11 and 14 months respectively. Starting with, $t_{\text{cutoff}} = 8$ months, the PIT histograms at all three forecast horizons showed a strong bias, with most mass on the right side of the histogram (indicative of underestimation). This behaviour was very similar to the PIT histograms at the earliest cutoff point in SI-type epidemics, although here the bias was clearly more pronounced.

The PIT histograms at $t_{\text{cutoff}} = 11$ months exhibited some overdispersion, as reflected by the slight $\cap$-shape. The histograms at $t_{\text{cutoff}} = 14$ months also exhibited some overdispersion, particularly for the 1 month ahead forecast. In the 6 months and 1 year ahead forecasts, the top 5% of the DGD was particularly underrepresented.

Chapter 7

Real-world data application

The present chapter applies the Bayesian Gamma GLM described previously to forecast using real-world data-set: Meningococcal disease in the Netherlands between 2012 and 2018.

7.1 Background

Meningococcal disease is caused by the bacteria *Neisseria meningitidis*. Approximately 10% of the general population carry the bacteria asymptomatically, but in a small proportion of cases it invades the bloodstream and can subsequently cause complications such as septicaemia or meningitis. For those infected, the disease progresses rapidly, leading to a high case fatality and long-term consequences for survivors, such as limb amputation and brain damage [Knol et al., 2017].

The strains (genetic variants) of the disease are classified into so-called serogroups: types A, B, C, W, X and Y are responsible for 90% of cases worldwide. Since 2012, the number of serogroup W cases (referred to as 'MenW') in the Netherlands has seen a large increase. In particular, there were 138 hospitalised cases between 2015-2017, with an associated fatality rate of 12% [Knol et al., 2018]. This sharp increase is clearly visualised in Figure 7.1 below.

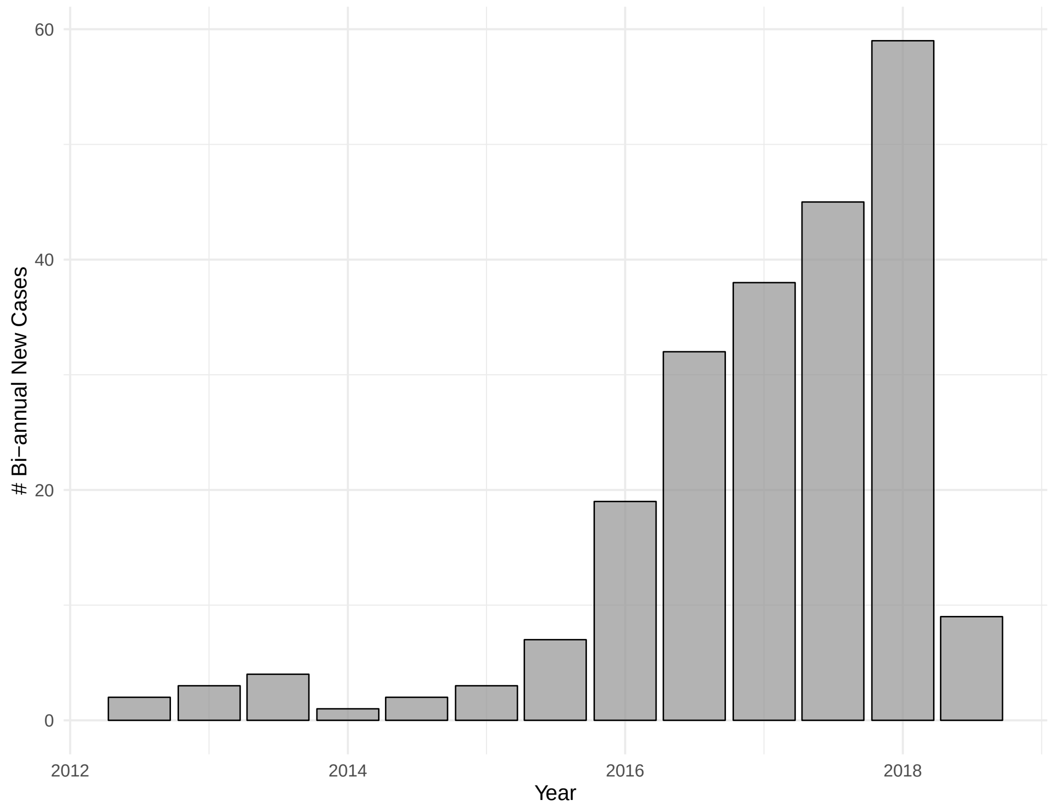

Figure 7.1: Number of new hospitalised MenW cases since 2012 on a bi-annual (6-month) basis. Note that the latter half of 2018 was not a sharp decrease: data was only available up until end of July 2018, and so few cases were reported between then and the end of June 2018.

As a result, a quadrivalent conjugate MenACWY vaccine has been introduced since May 2018 for 14 month-olds, and since October 2018 has also been offered to 13-14 year-olds. Carriage of the meningococcus does not confer immunity to subsequent carriage, and infection does not confer immunity against subsequent infections.

The latter statement makes the MenW outbreak a suitable candidate from which to forecast assuming SIS dynamics. The objective of the current section is to use the time between successive observed (hospitalised) cases to provide a forecast of future MenW hospitalisation incidence 2.5 years ahead (until the end of the year 2020), assuming no vaccine implementation. To do so, we will apply the Gamma GLM with inverse link using the tailored MH sampler, and forecast forward using Gillespie's direct method described previously. Furthermore, the Bayesian estimates will be compared to the frequentist estimates.

7.2 Data description

Data regarding MenW in the Netherlands was obtained from the Center for Epidemiology and Surveillance of Infectious Diseases, located at the Dutch National Institute for Public Health and the Environment. The data contained $n = 225$ hospitalised individuals (we refer to these as observed or recorded cases interchangeably), each with a recorded approximate date of infection provided by the treating physician.

Due to the rapid progression of the disease, this date is often close to the date of symptom onset. Nevertheless, we assumed it to be the date of infection for the present analysis. The first observed case was recorded on May 12^{th}, 2012; and the latest on July 28^{th}, 2018.

7.3 Model preparation: reconstructing number of infectious

In preparation for the Gamma GLM, the first step was to prepare the dependent variable y: time between successive observed cases. Specifically, we had $n - 1 = 224$ interevent times, which we recorded in days. If successive infections were recorded on the same day, we set the corresponding interevent time to 0.5, rather than 0, to avoid zeros in the analysis.

Second, we prepared the explanatory variable x. As Meningococcal infection follows SIS-type dynamics, we used the risk-set approximation of I as outlined in subsection 4.4.2. Reportedly, the average duration of meningococcus colonisation is approximately 11 months [De Wals and Bouckaert, 1985, Coen et al., 2000]. As such, we fixed the average duration of infectivity $\gamma^{-1} = 335$ days. Figure 7.2 (panel B) below visualises the newly created variable as a function of time.

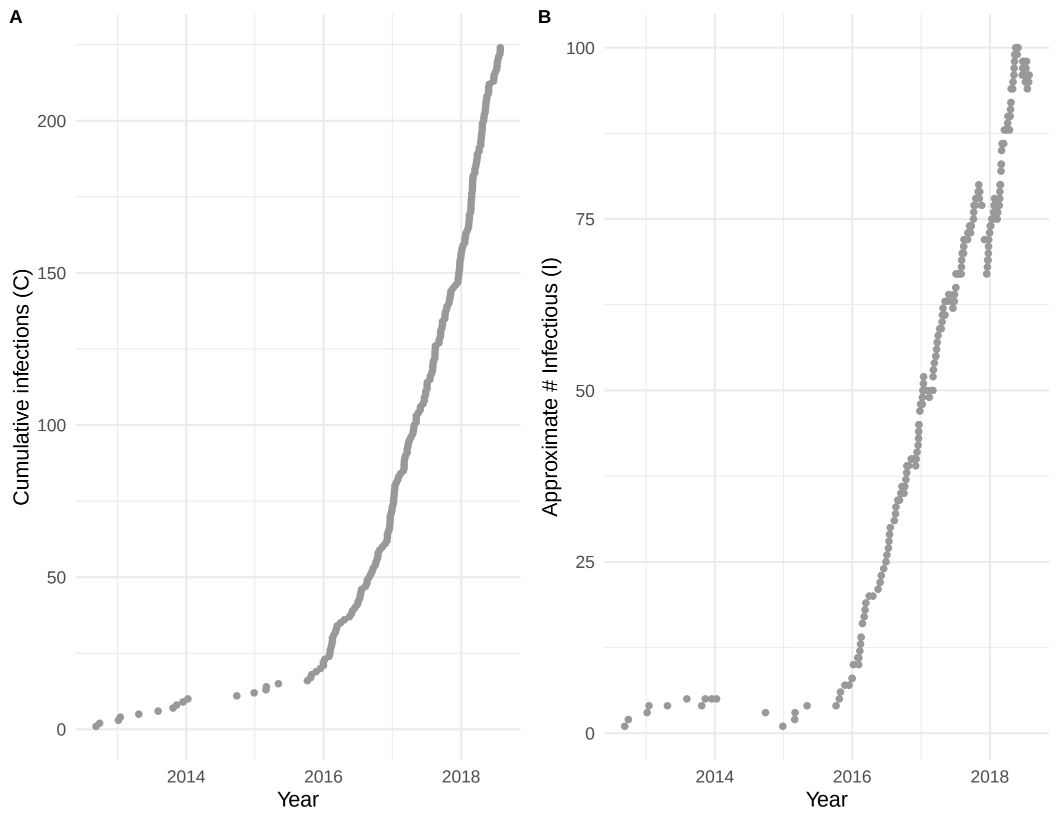

Figure 7.2: Panel A: cumulative counts of hospitalisation as a function of year. Panel B: risk-set approximation of I as a function of year.

For comparison, we also plotted the cumulative incidence of hospitalisation for the same data as a function of time (panel A). Despite the resemblance shape-wise to panel A, the new variable was not monotonically increasing. Each fluctuation (downwards) corresponds to a previously infected individual who eventually recovered and left the risk-set. It is worth refreshing that not all individuals survived in the MenW data, and hence wouldn't in this case reintegrate the susceptible pool. The variable assumed all patients survived the infection and eventually recovered, making it a clear *approximation* of I.

Furthermore, the interpretation of I requires clarification. Here, I corresponds to the number of hospitalised cases still infectious at a given time t, rather than all carriers of the disease at a given time. Implicit in this model thus is an assumption of a constant ratio between number of infections leading to hospitalisation, and number of new acquisitions of carriage.

7.4 Posterior diagnostics

Taking the time between successive observed cases to be the dependent variable, and the risk-set approximation of I to be the explanatory variable, a Bayesian Gamma GLM was run as per the model described in subsection 4.4.3. Using the tailored MH sampler, 4 parallel chains of 10 000 iterations were run, each with a 'burn-in' phase of $0.25 \times$ chain length. This implied 7500 iterations per chain, combining for a total of 30 000 samples from the posterior.

Concerning initial values, $\alpha^{(0)}$ was drawn randomly from $\mathcal{U}(0.5, 1.5)$, since the parameter space for was not expected to deviate much from that interval (where $\alpha = 1$ corresponds to exponential interevent times). The initial values for the regression coefficients $\boldsymbol{\beta}^{(0)}$ were the estimated coefficients from the weighted linear model in subsection 3.2.6. Note that for a different data set, there is no guarantee that these coefficients satisfy $0 < \boldsymbol{\eta}$: in this case, different starting values should be specified in order for the sampler to be able to evaluate the first iterations and then randomly walk towards the target distribution.

All chains had an acceptance rate just over 40%, which was a testament to how well the elements of the weighted linear model (variance-covariance matrix) came together to create an appropriate proposal distribution. Accordingly, the traceplots in panel A of Figure 7.3 unsurprisingly showed good mixing. Furthermore, Gelman and Rubin's convergence diagnostic was also computed. Resulting values were very close to 1 for all parameters, which again was a positive sign of convergence.

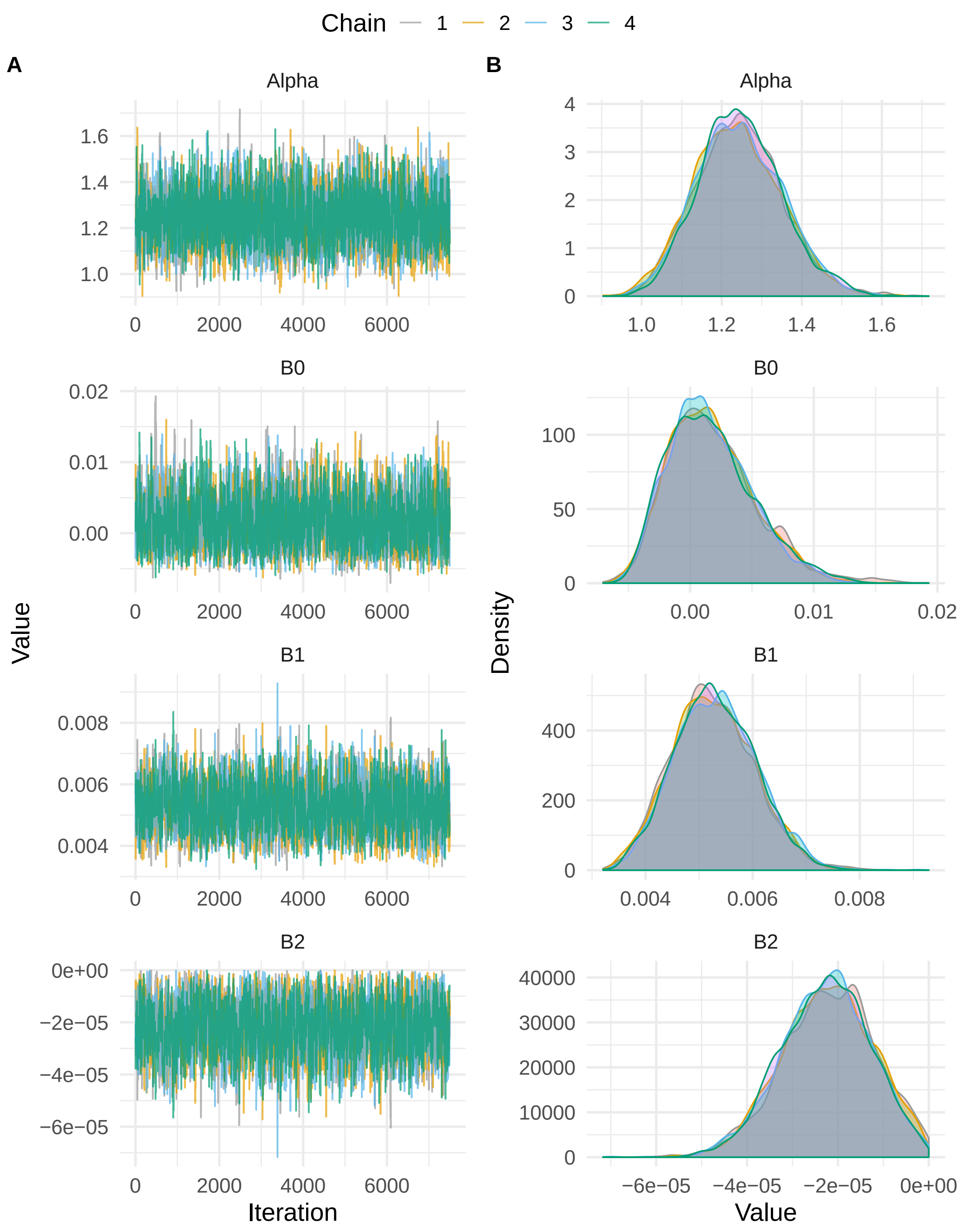

Figure 7.3: Panel A: traceplots for all model parameters. Panel B: resulting parameter densities, the chains are overlaid.

Lastly, a certain degree of autocorrelation between the samples in the chains was to be expected. The *effective sample size* gives us an idea of the number of ‘independent samples’ in a chain after taking autocorrelation into account. In the marginal posterior distribution for α, approximately 38.54% of the 7500 samples were independent, which was markedly more than for any of $\boldsymbol{\beta}$ (28.43%, 26.42% and 24.89%, for β_0, β_1 and β_2 respectively). Figure A.1 visualises the correlation between each posterior draw and it's k^{th} lag (for example, $k = 20$ correlation between iterations 100 and 80, 101 and 81, etc.), for all parameters and all chains. For good mixing, we would expect this correlation to decrease as k increases, which seemed to be the case in the figure.

7.5 Model fit

				90% Credible Interval	
	Mean	MAP	Median	5%	95%
β_0	$1.70 \cdot 10^{-3}$	$3.71 \cdot 10^{-4}$	$1.30 \cdot 10^{-3}$	$-3.25 \cdot 10^{-3}$	$8.03 \cdot 10^{-3}$
β_1	$5.28 \cdot 10^{-3}$	$5.35 \cdot 10^{-3}$	$5.25 \cdot 10^{-3}$	$4.08 \cdot 10^{-3}$	$6.57 \cdot 10^{-3}$
β_2	$-2.24 \cdot 10^{-5}$	$-2.36 \cdot 10^{-5}$	$-2.20 \cdot 10^{-5}$	$-3.91 \cdot 10^{-5}$	$-6.22 \cdot 10^{-6}$
α	1.24	1.27	1.24	1.07	1.42

Table 7.1: Summary of posterior quantities: mean, MAP, median and 90% credible interval. Note the MAP estimate is the mode of the annealed posterior distribution.

Table 7.1 above summarises the estimates obtained in the Bayesian Gamma GLM. The MAP estimates were obtained by running the sampler again with 4 chains of 10 000 iterations, this time using simulated annealing with geometric tempering constant $\alpha^t = 1.002$. The distributions for N, I^* and R_0 were obtained by multiplying/dividing the chains for $\boldsymbol{\beta}$ and α according to the estimators defined in subsection 2.3.2, and the estimates can be found in table 7.2.

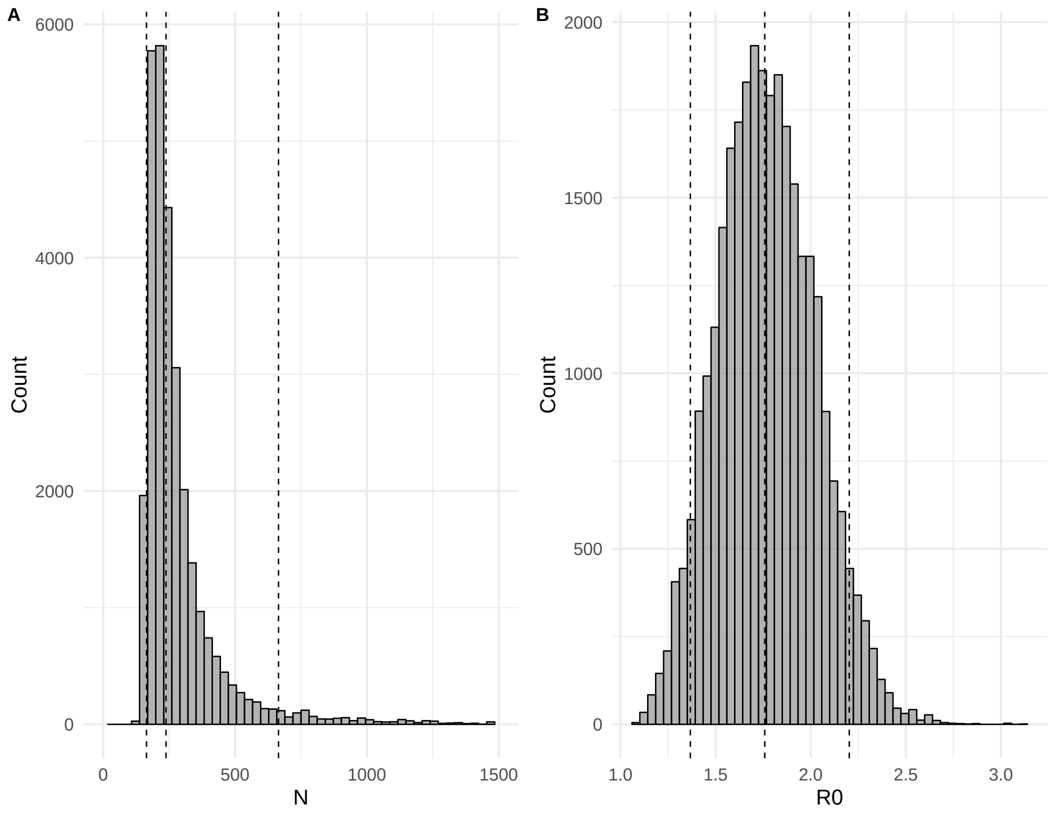

Figure 7.4: Posterior distributions of N (panel A) and R_0 (panel B). Dashed lines represent the 5^{th}, 50^{th} (median) and 95^{th} percentiles.

Of particular note were the distributions of N (the population size) and I^* (the endemic equilibrium), which were both clearly right-skewed. For N, we can visualise this clearly in Figure 7.4. Due to these distributions, we chose to report a 90% credible interval, which was more stable from run to run of the sampler. For these two distributions, the median and MAP estimates were also far more stable compared to the mean, which was easily affected by the skew of the distributions.

	Mean	MAP	Median	90% Credible Interval 5%	95%
N	401.00	227.02	238.00	163.99	664.71
I^*	137.96	100.28	102.78	82.89	188.31
R_0	1.77	1.79	1.76	1.37	2.20

Table 7.2: Posterior mean, MAP, median and 90% credible interval for N, I^* and R_0.

Focusing on the MAP estimates, we can see that $\beta_0 \approx 0$ (as expected from the data-generating model), despite not forcing the coefficient to zero. Furthermore, $\alpha = 1.27\,[1.07, 1.42]$, which was consistent with the expected $\alpha > 1$ for real-world infections. The estimated $R_0 = 1.79\,[1.37, 2.20]$ was higher compared to usually reported values for Meningococcal disease, which fluctuate between 1.40 and 1.60 [Hoek et al., 2018].

Figure 7.5 below plots the model fit using MAP esimated to observed interevent times. The median and mean fits were not plotted as the lines were mostly overlapping. The shape of the points seemed to follow a characteristic SIS-like pattern: the time between successive infections was large when few individuals were infectious, and became smaller as I increased.

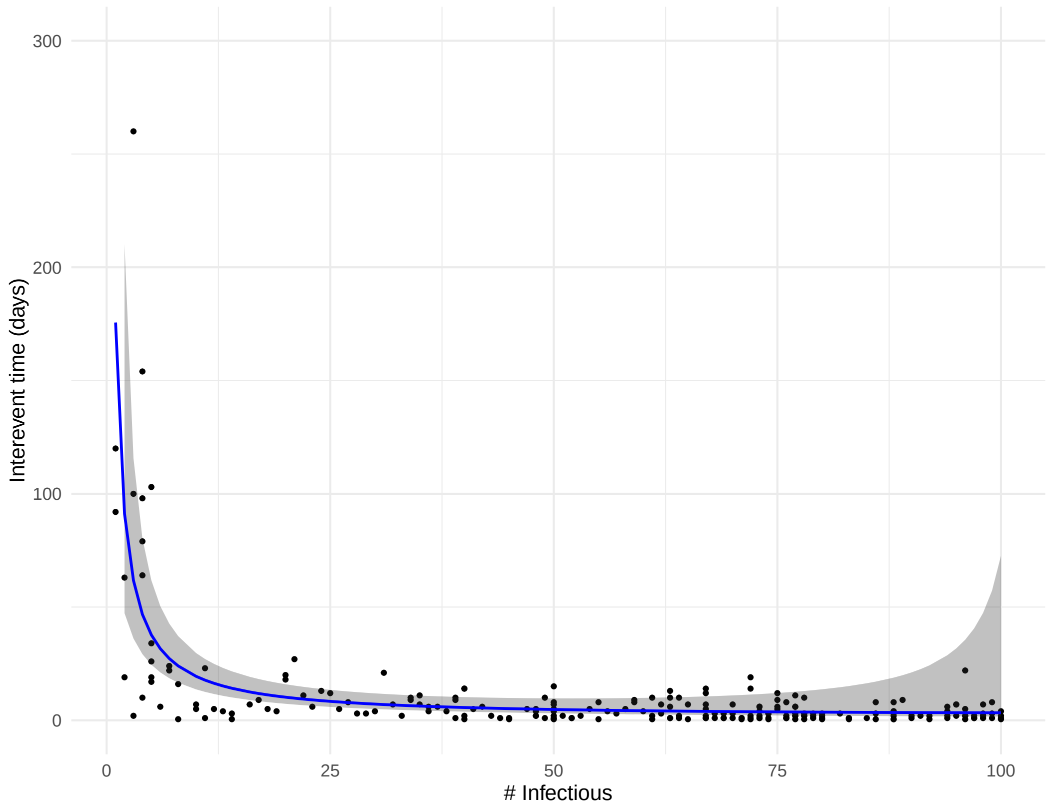

Figure 7.5: MAP fit (drawn line) and 90% credible interval (grey area). '# Infectious' was the explanatory variable x in the model, the interevent times were the dependent variable y. The median and mean fit were not plotted as the lines were mostly overlapping.

7.6 Comparison of Bayesian GLM fit to frequentist GLM fit

Since weakly informative (Cauchy and half-Cauchy) priors were used in the Bayesian approach, we would expect the MAP estimates to be very close to the estimates obtained via maximum likelihood in a frequentist GLM. Thus, a frequentist Gamma GLM was also run, and the corresponding estimates are shown in table 7.3 below. The MAP estimates were used as starting values for the GLM. In R, the GLM was implemented by specifying `formula = y ~ x1 + x2`, `family =`

`Gamma(link = "inverse")` and setting `start` equal to the vector of MAP coefficients.

				90% Confidence Interval	
	Estimate	SE	p-value	Lower	Upper
β_0	$5.23 \cdot 10^{-4}$	$3.37 \cdot 10^{-3}$	$8.77 \cdot 10^{-1}$	$-5.01 \cdot 10^{-3}$	$6.06 \cdot 10^{-3}$
β_1	$5.33 \cdot 10^{-3}$	$7.98 \cdot 10^{-4}$	$2.50 \cdot 10^{-11}$	$4.01 \cdot 10^{-3}$	$6.64 \cdot 10^{-3}$
β_2	$-2.30 \cdot 10^{-5}$	$1.05 \cdot 10^{-5}$	$2.81 \cdot 10^{-2}$	$-4.03 \cdot 10^{-5}$	$-5.77 \cdot 10^{-6}$
α	1.24	$1.05 \cdot 10^{-1}$	-	-	-

Table 7.3: Summary of GLM estimates, and 90% confidence interval. Note β_1 was statistically significant at the $\alpha_{H_0} = 0.001$ level, and β_2 at $\alpha_{H_0} = 0.05$ level. H_0 here being $\beta_j = 0$. There was thus evidence for a linear and quadratic term in the model. The approximate standard error for α was computed as the square-root of the reciprocal of the observed information.

As we can see, the MAP estimates were extremely similar to the GLM estimates (all terms were of the same order). Likewise, the credible and confidence intervals were of comparable width. Concerning the shape parameter, there are estimation procedures. The first is the MLE, as per Venables and Ripley [2002], by using the `gamma.shape()` function in their 'MASS' package in R. The second is the moment estimator proposed by McCullagh and Nelder [1989] as a result of suggesting the MLE was too sensitive to rounding errors of small values.

For the present data, the MLE was estimated at $\alpha = 1.24$, which was very close to the MAP estimate. In contrast, the moment estimator yielded $\alpha = 1.05$, which was substantially lower than the MAP estimate from the posterior distribution. Note that the choice of α estimator has important bearing on the standard errors of the coefficients. In table 7.3, the standard errors and associated confidence intervals are based on the MLE of α.

		90% Confidence Interval	
	Estimate	Lower	Upper
N	231.48	-	-
I^*	101.78	-	-
R_0	1.78	-	-

Table 7.4: As N, I^* and R_0 were computed as linear combinations of the estimated (mean) coefficients, the 90% confidence interval was not readily available. It could however be approximated by using Fieller's method [Fieller, 1954].

The frequentist estimates for N, I^* and R_0 are found in table 7.4 above. All 3 values were extremely close to the MAP estimates. As they were obtained by linear combinations of the estimated regression coefficients, approximate confidence intervals were not computed. If desired, they could be computed by Fieller's method [Fieller, 1954].

Futhermore, the residual deviance of the quadratic model was 203.03 on 221 degrees of freedom (df), and the null deviance was 458.79 on 223 df. The comparatively lower residual deviance is indicative of good model fit ($\chi^2(2) = 255.76, p < .001$). The chi-square tests for difference in model deviance were also statistically significant for the linear model compared to the null model ($\chi^2(1) = 251.90, p < .001$), and for the quadratic model compared to the linear model ($\chi^2(1) = 3.86, p = .049$).

Thus, we have performed a successful check of Bayesian GLM against standard GLM on the MenW dataset: the MAP estimates are very close to the frequentist estimates. In the frequentist paradigm, the model is justified based on statistically significant coefficient tests, and chi-square deviance tests. We have sufficient evidence for including both the linear and quadratic terms in the model.

7.7 Forecasts

To forecast 2.5 years ahead, 1000 draws were randomly sampled from the 30 000 total draws of the posterior distribution. The results are summarised in Figure 7.6. For a given epidemic, giving I at any a time t is not necessarily the most informative. Instead, we can sum up the transitions in the forecasts that resulted in infections over chosen time intervals, and summarise the number of new cases. We chose here to do this bi-annually, and took the median and 90% credible interval for the number of new cases. Panel A of the same figure seemed to show that the outbreak has reached an equilibrium of about 50-55 new observed cases per 6 months. Furthermore, the credible intervals became larger with time, particularly the upper bound (95th percentile).

Panel B visualises the forecasting from the perspective of I over time. Each coloured line corresponds to a single Gillespie run, based on a single, randomly sampled draw from the posterior. We refer to all 1000 forecasts together as the 'forecast cloud'. Furthermore, we also plotted the solution of the ordinal differential equations that govern SIS models, using the MAP (solid line), median (dashed line) and mean (dotted line) coefficients. These are deterministic forecasts. As we can see, the MAP and median coefficients overlapped substantially, both appearing stable around the $I = 100$ mark. In contrast, the deterministic mean forecast was far more pessimistic, forecasting $I > 150$, 2.5 years ahead. This was most likely a result of the skewed distribution of N.

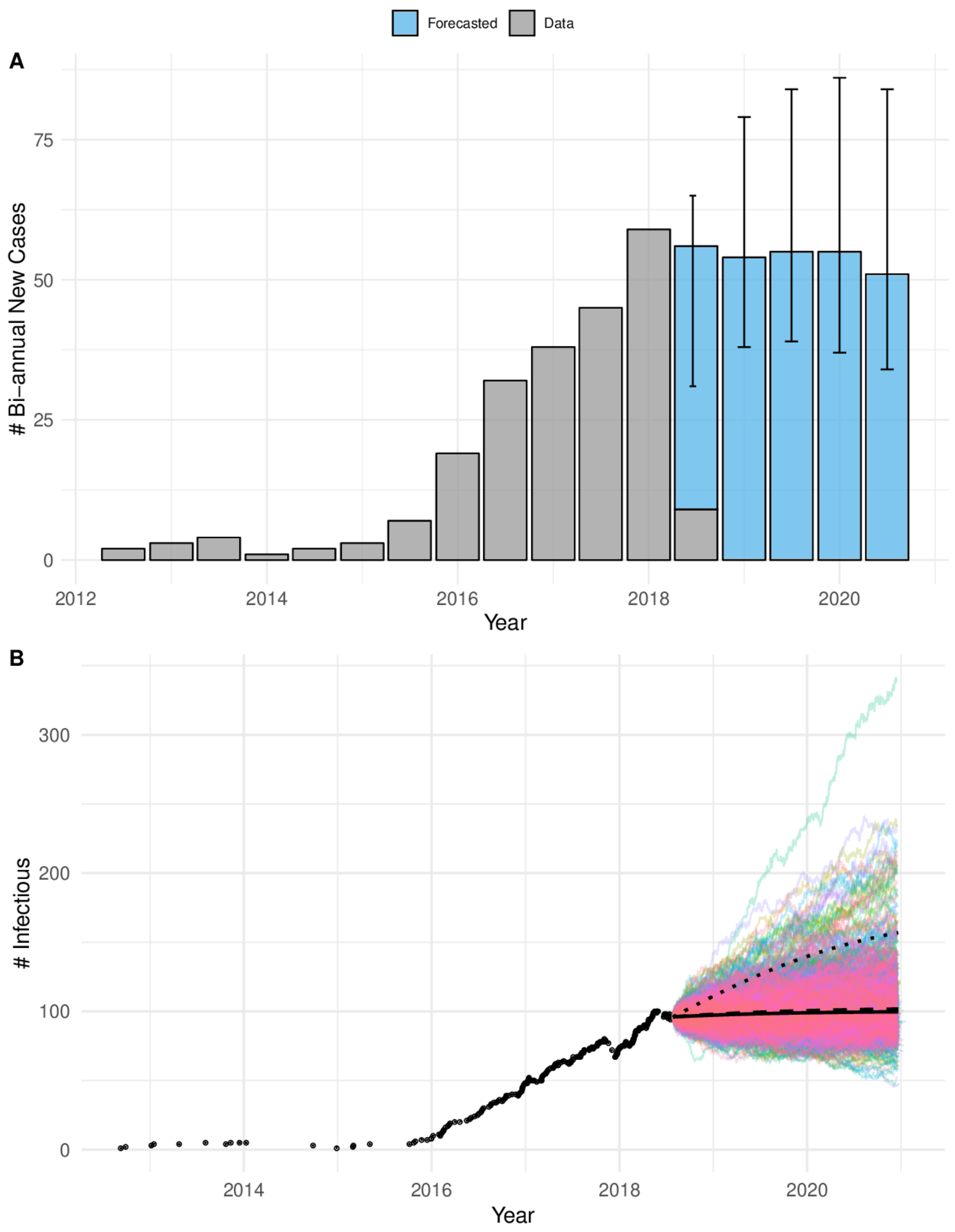

Figure 7.6: Visualisation of 1000 Gillespie forecasts for the MenW data. Panel A summarises the median bi-annual new observed cases (and associated 5^{th} and 95^{th} percentiles). Panel B visualises forecasts in terms of I over time. Deterministic forecasts also plotted: MAP = solid line, median = dashed line, mean = dotted line.

Chapter 8

Discussion

This final chapter provides a summary of results from the simulation experiment and the MenW analysis, and proceeds to provide avenues for future research.

8.1 Summary

8.1.1 Calibration of the method

A large-scale simulation experiment was run to evaluate the calibration of the forecasting technique for SI, SIR and SIS-type epidemics, using PIT histograms. This was done by cutting off simulated datasets at 3 different points, and forecasting to 1 month, 6 month and 1 year ahead horizons.

If a bias or over/undispersion was present, it was primarily dependent on the time of cutoff, and was accentuated the further the forecast horizon was. The clearest examples of this were the PIT histograms for the earliest cutoffs in SI and SIS epidemics, where there were clear hints of underestimation. This underestimation became more pronounced for the 6 month and 1 year ahead forecasts compared to the 1 month ahead forecast.

Another example of bias was found at latest cutoff for SIR-type epidemics, where the lowest 5% of the DGD was disproportionately well represented, indicating potential overestimation. This was the case at all three forecast horizons.

The remaining cutoff-horizon combinations were rather well calibrated, although a common theme remained: the extremes of the DGD were overall underrepresented, which was indicative of forecasts with too high variance (overdispersion).

Importantly, due to keeping simulated datasets with at least 25 infection events, the earliest cutoffs (for SI and SIS particularly) had comparatively fewer datasets from which to record PIT values. This may have in small part been responsible for the strong bias detected in these histograms. For the remaining histograms that did not suffer from this issue, any departures from uniformity outside of the extremes of the DGD were rather irregular. This most likely indicates that 1000 simulated datasets per cutoff-horizon pair was not enough, and the number of datasets needed for smoother PIT histograms may need to be larger by a factor of at least 10. Only then would we be able to draw more confident conclusions on the nature of any bias or over/underdispersion present.

Last, the number of forecasts performed for each dataset (100) may also have been too few, and was mainly chosen for computational reasons. Using a greater number (say, 1000) would yield a more detailed predictive distribution and by extension a more precise PIT value.

8.1.2 MenW forecasting

Assuming SIS-dynamics, a Bayesian Gamma GLM was run on the time between successive observed (hospitalised) cases on MenW data in the Netherlands between spring 2012 and summer 2018. A comparison between the Bayesian MH sampler and the frequentist GLM was also performed. The MAP estimates were very close the the maximum likelihood estimates, and confidence/credible intervals were of comparable width.

The MAP basic reproductive number R_0 was estimated at approximately 1.79 secondary cases per primary case; higher than usual values reported in the literature. The estimated MAP endemic equilibrium was approximately $I^* = 102.31$ hospitalised MenW cases that are infective. In other words, the number of infectious observed cases at any given time. Specifically in this model, it was the number of asymptomatic carriers multiplied by an implicitly assumed constant case-to-carrier ratio.

The posterior distribution was then used to forecast stochastically 2.5 years ahead, using the Gillespie's direct method. It would seem as if the MenW outbreak in the Netherlands was (from August 2018 onwards) at or near an equilibrium of approximately 50-55 new observed cases per 6 months, with credible intervals increasing over time.

In light of the PIT histograms for SIS-type epidemics, interpretation of the MenW forecast depends on whether or not we believe the epidemic was in an early growth phase, or whether it was much later in the growth phase. In case of the former, the PIT histograms would seem to suggest that this forecast is a rather strong underestimation of future MenW infective hospitalised cases. In the case of the latter, the variance in forecasts may still be too high to make any meaningful conclusions. Due to this large variance, it is unclear whether the conservative (in the case of MenW) MAP and median estimates are better for prediction compared to the more pessimistic mean forecast.

8.2 On the use of interevent times

In the literature concerning infectious disease forecasting, it would seem that until now the use of infection interevent times in conjunction Bayesian methods is absent. This is rather surprising considering the simplicity of the data required: only a single vector of infection dates is needed. The cumulative counts are then easily created with a vector containing a sequence from 1 to n.

Nevertheless, individual-level data is required, which may be an issue to obtain in outbreaks with stronger transmission (resulting in many infections recorded on the same day). These could result in binned data, which presents a different analytical challenge. Namely, we may have to resort to jittering the binned times in order to mimic individual infection events.

Even with individual-level data, other issues remain present. In the MenW case, the date of infection was estimated by the physician. This date becomes increasingly uncertain, the slower the progression of a particular disease. In even less ideal conditions, we could encounter interval censored data, where the infection is only known to have occurred in a certain interval of time.

8.3 Bayesian approach and alternatives

The Bayesian approach to forecasting in this thesis possesses some clear advantages. First, the likelihood-related constraints of the model are relatively straightforward and flexible to implement. Here, this was done via rejection sampling of the proposal distribution in the MH sampler. Second, the forecasts incorporate uncertainty in the parameter values. Importantly, this allows us to create a detailed (dependent on number of posterior samples) predictive distribution at different points in time, and subsequently evaluate calibration. Third, the hope is that researchers opt to use Bayesian methodology more often, and we can gain progressively more evidence to better inform the choice of prior distributions.

Indeed, the Bayesian approach is particularly attractive compared to its frequentist counterpart, which may often fail to converge if appropriate starting values aren't provided. This potential failure to converge affects procedures such as bootstrapping which become unstable as a result [Verkerk, 2018]. The Bayesian Gamma GLM with weakly informative priors could be considered as a type of 'robust GLM': an alternative to the frequentist approach that does not fail to converge. Section 7.6 seems to support this claim, suggesting it is a very good approximation to the ordinary frequentist GLM.

In addition, with the Bayesian approach, credible intervals for quantities such as N and R_0 (obtained from linear combinations of the regression coefficients) are readily available; while in the frequentist paradigm they can only be approximated using Fieller's method or the Delta method. This allows us to re-emphasise the crucial advantage of the Bayesian approach: working with the posterior predictive distribution. In the frequentist paradigm, outside of bootstrapping, there seems to be no straightforward way to obtain and then work with a predictive distribution at different points in time. By extension, this implies calibration is difficult to evaluate.

However, an advantage of the frequentist approach concerns the hypothesis testing of regression coefficients, and the deviance tests. This allows us to make inference about whether or not we have enough evidence to include the linear and quadratic terms in our model. This is rather important, as it could give an indication as to a potential mismatch between the data-generating model and the regression model. In a Bayesian paradigm, Bayes factors may provide an equivalent solution by quantifying the support of one model over another (e.g. quadratic model over the linear model) through ratios of posterior and prior probabilities [Kass and Raftery, 1995]. This clearly needs to be explored further to make the comparison between the Bayesian Gamma GLM and its frequentist alternative more complete.

An alternative approach involves the use of penalised regression, using ridge, LASSO or elastic-net penalties on the regression coefficients [Hastie et al., 2009]. The only piece of software in R currently implementing penalised regression for a Gamma GLM with inverse link is the 'h2o' package [Mark Landry, 2017]. Even though the penalising may aid in estimating coefficients, the trade-off of bias over variance means we have no clear way of establishing meaningful confidence intervals, and thus a predictive distribution.

8.4 Interactions, effects of covariates

The models outlined in this thesis should in theory work well, *provided* that the data-generating models are correct. In the cases where they are not, it is worth including additional covariates in the model to see whether predictions are improved, or whether any informative interactions exist.

Consider a disease with SIS-type dynamics, where there is some evidence of *seasonality*: there may be more cases in some months of the year relative to others. An immediately relevant covariate here may be weekly/monthly temperatures (degrees celsius). Rather than the marginal effect of the covariate itself on the infection rate, we may be more interested in its interaction with the infection parameter β, or β_1 in GLM terms. This interaction term could inform, for example, whether there is increased transmission in the colder months of the year relative to the warmer ones.

Including said covariate and associated interaction in the Gamma GLM with inverse link, using the tailored MH sampler depends on the coding of the covariate. Continuing with the same example, if we keep temperature as a continuous covariate we simply need to multiply the posterior samples of β_1 and β_{Temp}, and use the resulting samples to make inference. If however we decided to code temperature as a binary variable (e.g. 0 for the six warmest months of the year, and 1 for the coldest six), this would amount to sampling two sets of parameters: one for each level of the binary variable. This discrepancy between the posterior distribution β_1 in the colder and then in the warmer months would thus be of interest.

8.5 Open questions

Open questions regarding the Bayesian approach still remain. First, we know we can provide a detailed predictive distribution at different points in time, but which forecast we should *trust* for prediction is still unclear. We can opt for the more stable median and MAP estimates, or we could use the more pessimistic (skewed) mean estimate. Furthermore, how we choose the 90% credible interval is also of interest: knowing that the distribution of N is very skewed upwards, the 0^{th} and 90^{th} percentiles may be more useful for prediction.

On the same point of prediction, the extent to which we need to know about the data-generating model (infection cycle) is unclear. The initial condition of an epidemic, the under-reporting rate ρ^{-1} and the population size N may or may not all be necessary to make good predictions. Similarly, it is unclear whether knowledge of the cumulative counts C_t is enough to make good predictions, or whether we may need to approximate I somehow like in the SIS-case.

Furthermore, future research must focus on better ways of specifying prior distributions for case of a Gamma GLM with inverse link. In particular, it would be useful to move away from weakly informative priors to more informative ones, depending on the disease in question. For example, if we know a disease to have particularly high transmission rate, we could impose a stronger prior on β_1 away from zero. This would result in sharper prediction intervals for number of future infections.

Appendix A

Extra figures MenW results

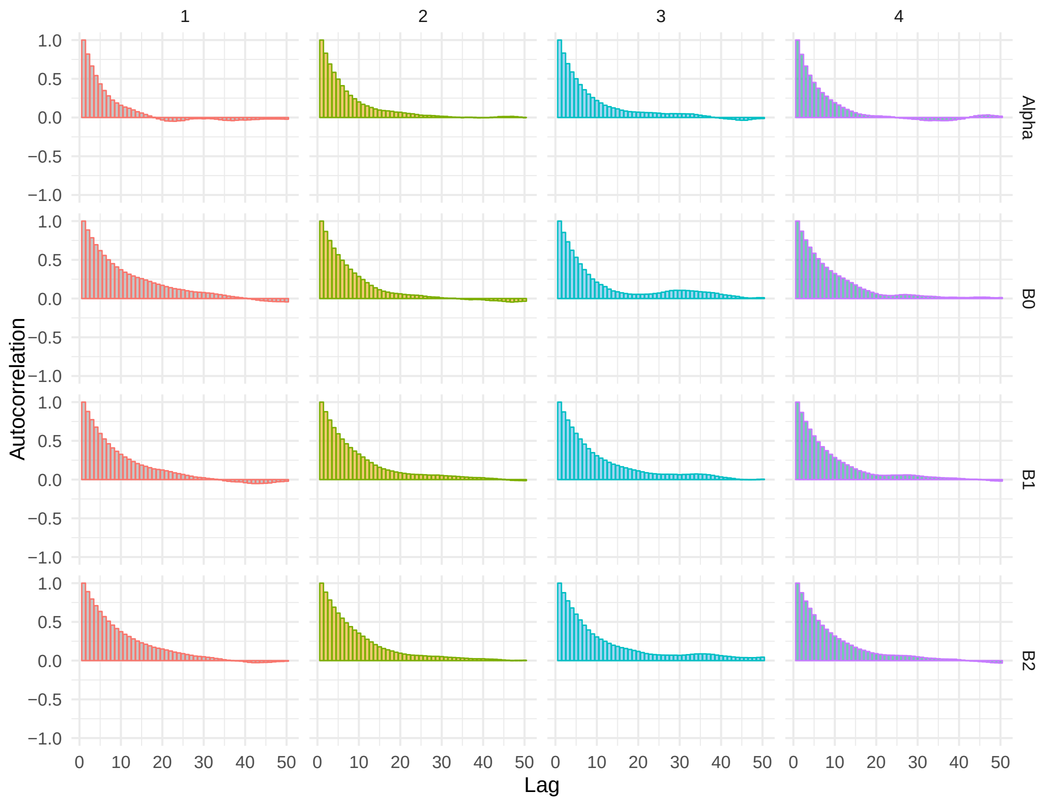

Figure A.1: Autocorrelation plots per chain for each parameter in the MH sampler.

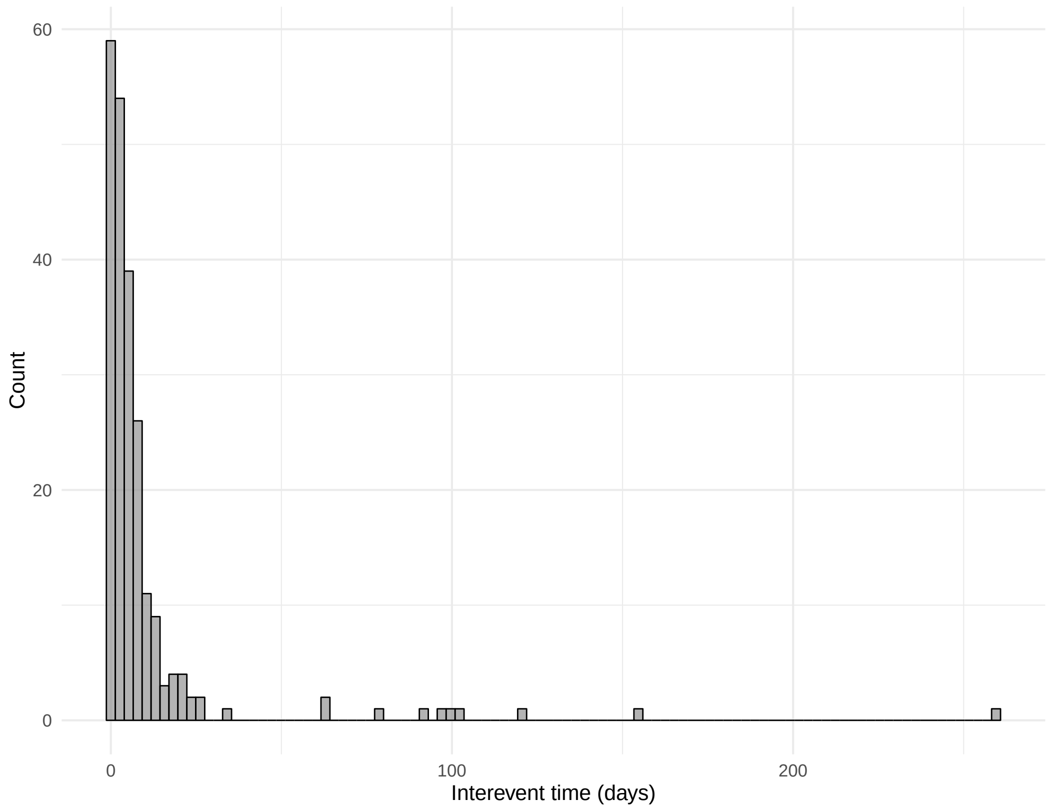

Figure A.2: Distribution of reported infection interevent times (days) for MenW in the Netherlands.

Appendix B

R-code

This appendix contains all the R-code used in the thesis. Parts of the code require external files (e.g. MenW dataset, .csv files containing PIT simulations) to run. These may be available upon request.

B.1 Packages used

Several R packages were used in this manuscript. A list with corresponding citations, and main use, is found below:

- 'foreign' - R Core Team [2017].
  Allowed reading of SPSS files.

- 'MASS' - Venables and Ripley [2002].
  Used to compute the MLE of the shape parameter from the Gamma GLM, and to generate random numbers from a multivariate normal distribution.

- 'survival' - Therneau [2015].
  Used to create the risk-set approximation of I in the SIS model.

- 'deSolve' - Soetaert et al. [2010].
  Differential equation solver that facilitated deterministic forecasts.

- ‘plyr’ - Wickham [2011].
  Various data-wrangling operations.
- ‘tidyverse’ - Wickham [2017].
  Essential collection of packages for data manipulation, of which ‘ggplot2’ was used for all plots.
- ‘ggpubr’ - Kassambara [2018].
  Enabled multi-panel plots.
- ‘lubridate’ - Grolemund and Wickham [2011].
  Facilitated manipulation of dates.
- ‘anytime’ - Eddelbuettel [2018].
  Facilitated manipulation of dates.
- ‘ggmcmc’ - Fernández-i Marín [2016].
  Permitted ggplot2-based analysis and visualisation of MCMC chains (e.g. traceplots, posterior densities).
- ‘coda’ - Plummer et al. [2006].
  Provided multiple diagnostic tools for MCMC chains, such as effective sample size or the Gelman-Rubin diagnostic.
- ‘xtable’ - Dahl et al. [2018].
  For easy conversion of R table to LaTeXformat.

B.2 Support functions

The below are a collection of support functions created to simplify the reading of main code. They were called by using `source("UsefulFunctions.R")`, prior to running the chapter-by-chapter code.

```
### Data wrangling functions ~


# 1) Convert SPSS dates to R.
```

```
spss2date <- function(x) as.Date(x / 86400, origin = "1582-10-14")

# 2) Computes time since first infection in days.
# - Based on date vector of class "date"
time_since <- function(date_vector) {
  time <- as.numeric(difftime(date_vector, date_vector[1],
                              units = "days"))
  return(time)
}

# 3) Computes interevent times in days.
# - time_var is of the form returned from time_since function
inter_event <- function(time_var) {
  times <- (c(time_var, 0) - c(time_var[1], time_var))
  times <- ifelse(times == 0, 0.5, times)
  return(times[-length(times)])
}

# 4) Risk-set approach to I, data frame must have variables:
# - time = (approx.) time of infection
# - end = (approx.) end of infection = time + av. durat. colonisation
# - status = simply vector of 1s
infect <- function(data) {
  fit <- survfit(Surv(time, end, status) ~ 1, data)
  table1 <- cbind.data.frame(time = fit$time, I = fit$n.risk)

  # When I = 0, replace by 1
  table1$I <- ifelse(table1$I == 0, 1, table1$I)

  # Find closest time
  index <- sapply(data$time, function(x){
    unique(which.min(abs(table1$time - x)))
  })
  x1 <- table1[index, ]$I
  return(x1)
}

# 5) Getting roots of quadratic equation
# Constructing quadratic Formula
quad <- function(b0, b1, b2) {

  # Delta
  delta <- function(b0, b1, b2) {
    b1^2 - 4 * b2 * b0
  }

  # Find roots
  if(delta(b0, b1, b2) > 0) {
    x_1 <- (-b1 + sqrt(delta(b0, b1, b2))) / (2 * b2)
    x_2 <- (-b1 - sqrt(delta(b0, b1, b2))) / (2 * b2)
    return(c(x_1,x_2))
  }
  else if(delta(b0, b1, b2) == 0) {
    return(-b1 / (2 * b2))
  }
  else return(NA) # root doesn't exist
}

# 6) Mode
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

#**********************************************************************#

### Bayesian support functions ~
```

```r
# 1) Half-Cauchy Density - to use in posterior
dhcauchy <- function(x, sc){(2 * sc) / (pi * (sc^2 + x^2))}

# 2) Take array and make into ggmcmc object for nice visuals
# - also keeps the useful coda object
ggpost_convert <- function(post) {

  coda_obj <- as.mcmc.list(lapply(alply(post, 2), mcmc))

  # Make parameter labels
  P <- data.frame(Parameter = c("1", "2", "3", "4"),
                  Label = c("B0", "B1", "B2", "Alpha"))

  # Create ggs object
  ggs_obj <- ggs(coda_obj, par_labels = P)
  return(list(ggmc = ggs_obj, coda_ob = coda_obj))
}

# 3) Makes the 3-dim array of 4 chains into 1 data-frame
# - Makes computations easier
full_dens <- function(post) {

  dens <- apply(post, 2, function(x){
    x <- as.data.frame(x)
    names(x) <- c("B0", "B1", "B2", "Alpha")
    Sig <- 1 / 335 # for MenW case
    x$N <- -x$B1/x$B2
    x$I_star <-  x$N * (1 - 1 / x$B1 * Sig)
    return(x)
  })

  dens <- as.matrix(do.call(rbind, dens))
  return(dens)
```

```
}

# 4) Calculate deterministic function - for SIS
# Differential Equations
determ_func_SIS <- function(initial_values,
                            timepoints,
                            parameter_list) {

  sis_model <- function(current_timepoint,
                        state_values,
                        parameters) {
    # Define states
    S <- state_values[1] # susceptible
    I <- state_values[2] # infectious

    with(as.list(parameters), {

        # Get derivatives
        dS <- -beta * S * I + gamma * I
        dI <- beta * S * I - gamma * I

        results <- c(dS, dI)
        list (results)
      }
    )
  }

  # Format output
  output <- lsoda(initial_values,
                  timepoints,
                  sis_model,
                  parameter_list)

  output <- as.data.frame(output)
  output$Date <- as.Date(as.numeric(output$time),
                         # First MenW one
                         origin = as.Date("2012-05-12"))
  return(output)
}

# Source: http://rpubs.com/docblount/111138

# 5) Calculate deterministic function - for SI
determ_func_SI <- function(initial_values,
                           timepoints,
                           parameter_list) {



  si_model <- function(current_timepoint,
                        state_values,
                        parameters) {

    S <- state_values[1]
    I <- state_values[2]

    with(as.list(parameters), {

        dS <- -beta * S * I
        dI <- beta * S * I

        results <- c(dS, dI)
        list (results)
      }
    )
  }

  output <- lsoda(initial_values,
                  timepoints,
                  si_model,
                  parameter_list)

  output <- as.data.frame(output)

  output$Date <- as.Date(as.numeric(output$time),
                         # First MenW one
                         origin = as.Date("2012-05-12"))
  return(output)
}

# 6) Calculate deterministic function - for SIR
determ_func_SIR <- function(initial_values,
                            timepoints,
                            parameter_list) {

  sir_model <- function(current_timepoint,
                        state_values,
                        parameters) {

    S <- state_values[1]
    I <- state_values[2]

    with(as.list(parameters), {

        dS <- -beta * S * I
        dI <- beta * S * I - gamma * I
        dR <- gamma * I

        # combine results
        results <- c(dS, dI, dR)
        list(results)
      }
    )
  }

  output <- lsoda(initial_values,
                  timepoints,
                  sir_model,
                  parameter_list)
  output <- as.data.frame(output)
  output$Date <- as.Date(as.numeric(output$time),
                         # First MenW one
                         origin = as.Date("2012-05-12"))
  return(output)
}
```

B.3 Data simulation

The below contains the R-code for Gillespie's direct method for stochastic simulation of epidemics. By calling `source("DataSimulation.R")`, it is possible to simulate stochastically for SI, SIR and SIS-type epidemics.

```
### SIS Simulation (Gillespie) ~

# Function for all rates
rates <- function(B, Sig, N, I) {
  inf_rate <- B * I - (B * I^2) / N
  rec_rate <- Sig * I
  tot_rate <- rec_rate + inf_rate
  ratio_rate <- rec_rate / tot_rate

  # Return in easy to access vector
  rates <- c("Infect" = inf_rate, "Recov" = rec_rate,
             "Total" = tot_rate, "Ratio" = ratio_rate)
  return(rates)
}
```

```
# Gillespie code
gillespie <- function(c, # current event counts
                      t_origin, # start time
                      t_end, # end time (days)
                      n_steps, # max steps
                      B, # Beta
                      Sig, # Sigma
                      N, # Pop Size
                      I_start, # I begin
                      alpha) { # Current state

  samps <- matrix(NA, ncol = 4, nrow = n_steps)
  colnames(samps) <- c("Counter", "time", "I", "Transition")
  samps[1, ] <- c(c, 0, I_start, 0)

  # Initialize time and iteration
  i <- 1
  t <- 0

  while(i < n_steps - 1 && t < t_end) {

    # Current state and counter
    I <- samps[i, 3]
    c <- samps[i, 1]

    # In case I falls below 1
    if(I < 1){I <- 1}
    rate_iter <- rates(B, Sig, N, I)

    # Execute transitions
    if(runif(1) > rate_iter[4]) {
      samps[i + 1, 1] <- c + 1
      samps[i + 1, 3] <- I + 1
      samps[i + 1, 4] <- 1
    } else{
      samps[i + 1, 1] <- c
      samps[i + 1, 3] <- I - 1
      samps[i + 1, 4] <- 0
    }

    # Add interevent time
    t <- samps[i, 2]
    tau <- rgamma(1, shape = alpha, rate = alpha * rate_iter[3])
    samps[i + 1, 2] <- t + tau

    # Set counters
    i <- i + 1
  }

  # Format simulations
  samps <- as.data.frame(samps[complete.cases(samps), ])
  samps$Date <- as.Date(samps$time, origin = as.Date(t_origin))
  samps$S <- N - samps$I

  # To use in Metropolis
  samps$y <- inter_event(samps$time)
  samps$x1 <- samps$I
  samps$x2 <- (samps$I)^2
  return(samps[-1, ]) # Remove first row (interevent non existent)
}

#**********************************************************************#

### Gillespie for SI Epidemic ~

gillespie.SI <- function(t_origin, # start date
                         B, # Beta
```

```
                        N, # Pop Size
                        I_start,
                        alpha) { # Current state

  n_steps <- N - I_start # Until everyone gets infected
  samps <- matrix(NA, ncol = 2, nrow = n_steps)
  colnames(samps) <- c("time", "I")
  samps[1, ] <- c(0, I_start)

  # Initialize time and iteration
  for(i in 1:(n_steps - 1)) {
    I <- samps[i, 2]
    inf_rate <- B * I - (B * I^2) / N
    samps[i + 1, 2] <- I + 1

    # Compute time to next event
    t <- samps[i, 1]
    tau <- rgamma(1, shape = alpha, rate = alpha * inf_rate)

    # Update time and no. infected
    samps[i + 1, 1] <- t + tau
    samps[i + 1, 2] <- I + 1
  }

  # Format simulations
  samps <- as.data.frame(samps)
  samps$Date <- as.Date(samps$time, origin = as.Date(t_origin))
  samps$S <- N - samps$I

  # To use in Metropolis
  samps$y <- inter_event(samps$time)
  samps$x1 <- samps$I
  samps$x2 <- (samps$I)^2
  return(samps[-1, ]) # Remove first row (interevent non existent)
```

```
}

#*******************************************************************#

### SIR Gillespie ~

rates.SIR <- function(B, Sig, N, S, I) {
  inf_rate <- B * S * I / N
  rec_rate <- Sig * I ## check this
  tot_rate <- rec_rate + inf_rate
  ratio_rate <- rec_rate / tot_rate

  # Return in easy to access vector
  rates <- c("Infect" = inf_rate, "Recov" = rec_rate,
             "Total" = tot_rate, "Ratio" = ratio_rate)
  return(rates)
}

gillespie.SIR <- function(t_origin, # start date
                          t_end, # end time (days after t0)
                          B, # Beta
                          Sig, # Sigma
                          N,
                          S_start,
                          I_start,
                          alpha) { # Current state

  n_steps <- 10000 # max steps to take
  samps <- matrix(NA, ncol = 4, nrow = n_steps + 1)
  colnames(samps) <- c("Transition", "time", "S",  "I")
  samps[1, ] <- c(0, 0, S_start, I_start)

  # Initialize time and iteration
  i <- 1
```

```r
t <- 0

while(i < n_steps - 1 && t < t_end) {

  # Current state
  S <- samps[i, 3]
  I <- samps[i, 4]

  # In case I falls below 1, set I to 1
  if(I < 1){I <- 1}
  rate_iter <- rates.SIR(B, Sig, N, S, I)

  # Execute transitions
  if(runif(1) > rate_iter[4]) {
    samps[i + 1, 3] <- S - 1
    samps[i + 1, 4] <- I + 1
    samps[i + 1, 1] <- 1
  } else {
    samps[i + 1, 3] <- S
    samps[i + 1, 4] <- I - 1
    samps[i + 1, 1] <- 0
  }

  # Add interevent time
  t <- samps[i, 2]
  tau <- rgamma(1, shape = alpha, rate = alpha * rate_iter[3])
  samps[i + 1, 2] <- t + tau

  # Set counters
  i <- i + 1
}

# Format simulations
samps <- as.data.frame(samps[complete.cases(samps), ])
```

```
  samps$Date <- as.Date(samps$time, origin = as.Date(t_origin))

  # To use in Metropolis
  samps$R <- N - samps$S - samps$I
  samps$y <- inter_event(samps$time)
  samps$x1 <- samps$I
  samps$x2 <- (samps$I)^2
  return(samps[-1, ]) # Remove first row (interevent non existent)
}
```

B.4 Samplers

Here we present the R-code for the tailored MH sampler. First, we present the code exactly reflecting algorithm 1 in subsection 3.2.6, which involves rejection sampling the proposal distribution. Second, we present the MH sampler that is adaptive during burn-in. In this adaptive sampler, we don't rejection sample the proposal distribution, but equivalently set the acceptance probability to zero if a likelihood condition is violated.

Important: the adaptive part of the sampler can be disabled by setting `acc = NULL` in the function arguments, making it equivalent to the sampler using rejection sampling. For all analyses in the thesis, the function `MH_adap` was used as the main sampler with the option `acc = NULL`.

B.4.1 Tailored MH Sampler

```
# Proposal function
proposal <- function(params, var, X) {

  # Constrain eta > 0
  pos_linpred <- FALSE
  while(!pos_linpred) {
    # Betas
    propos <- mvrnorm(1, mu = params[1:3], Sigma = var)

    # Set bounds
    b1 <- propos[2]
    b2 <- propos[3]
    ratio <- - b1 / b2
    linpred <- X %*% propos
    alpha <- rlnorm(1, mean = log(params[4]), sd = 0.1)

    # b1 <= 0 or else either b1 or b2 could be negative
    pos_linpred <- !(any(linpred <= 0) | ratio <= 0 | b1 <= 0)
  }
  return(c(propos, alpha))
}

# Function computing log-posterior
post <- function(params, X, y) {

  # Likelihood
  alph <- params[4]
  betas <- params[1:3]
  lin_pred <- X %*% betas
  lik <- sum(dgamma(y, shape = alph, rate = alph * lin_pred, log = T))

  # Priors
  prior_B <- dcauchy(betas, 0, c(10, 2.5, 2.5), log = T)
  prior_alph <- log(dhcauchy(alph, 20))

  loglik <- sum(lik, prior_B, prior_alph)
  return(loglik)
}

# Function for picking initial values
```

```r
initial <- function(initial, dat, X, y) {

  # Generate a random (bounded) shape value
  alph <- runif(1, 0.5, 1.5)

  # Weighted lin mod
  lm1 <- glm(1/y ~ x1 + x2, weights = y^2,
             data = dat, family = gaussian())
  betas0 <- c(lm1$coefficients, alph) # Option 1
  initial <- c(initial, alph) # Option 2

  # Tests
  test1 <- suppressWarnings(!is.na(post(betas0, X, y)))
  test2 <- suppressWarnings(!is.na(post(initial, X, y)))

  if(test1) {
    return(betas0)
  } else if(test2) {
    return(initial)
  } else {
    stop("Pick some better starting values!
         They need to satisfy X %*% values > 0")
  }
}

# MH sampler
metropolis.gamma.trunc <- function(y, X,
                                   iters = 4000,
                                   start,
                                   nchains = 4,
                                   burn.in = 0.25,
                                   factor = (2.38^2) / 3,
                                   dat) {
```

```
# Find covmat of proposal
lm1 <- glm(1/y ~ x1 + x2, weights = y^2,
           data = dat, family = gaussian())
var <- vcov(lm1) * factor

# Storing beta samples
post_samples <- array(NA, dim = c(iters + 1, nchains, ncol(X) + 1))
accept <- array(NA, dim = c(iters + 1, nchains))

# For burn-in
burn <- floor(burn.in * nrow(post_samples)) + 1

# Initialize chains
for(j in 1:nchains) {
  # Initialize iterations
  for (i in 1:iters) {

    # Starting values
    post_samples[1,j,] <- start

    # Proposal
    x_tmin1 <- post_samples[i,j,]
    prop <- proposal(x_tmin1, var, X)

    # Correction factor
    num_c <- dlnorm(x_tmin1[4], log(prop[4]), 0.1, log = T)
    denom_c <- dlnorm(prop[4], log(x_tmin1[4]), 0.1, log = T)

    # Probabilities
    proposal_prob <- post(prop, X, y) + num_c
    current_prob <- post(x_tmin1, X, y) + denom_c
    prob <- exp(proposal_prob - current_prob)

    if (runif(1) < prob) {
        post_samples[i+1,j,] <- prop # Accept
        accept[i,j] <- 1
      } else {
        post_samples[i+1,j,] <- x_tmin1 # Reject
        accept[i,j] <- 0
      }
    }
  }

  accept <- accept[-nrow(accept),]

  if(!is.null(burn.in)) {
    final_samps <- post_samples[burn:nrow(post_samples),,]
    accept <- accept[burn:nrow(accept),]
    accept <- colSums(accept) / nrow(accept)
    return(list(post = final_samps, accept = accept))

  } else {
    accept <- colSums(accept) / nrow(accept)
    return(list(post = post_samples, accept = accept))
  }
}
```

B.4.2 MH with adaptive burn-in

```
# Initial values - inspired from rejection sampling
init <- function(params, var, X) {
  pos_linpred <- FALSE
  while(!pos_linpred) {
    propos <- mvrnorm(1, mu = params[1:3], Sigma = var)
    ratio <- - propos[2] / propos[3]
    linpred <- X %*% propos
    pos_linpred <- !(any(linpred <= 0) | ratio <= 0)
  }
  return(propos)
}

# Proposal
proposal <- function(params, var, X) {
  propos <- mvrnorm(1, mu = params[1:3], Sigma = var)
  alpha <- rlnorm(1, mean = log(params[4]), sd = 0.1)
  return(c(propos, alpha))
}

# Log-posterior
post <- function(params, X, y) {

  # Likelihood
  alph <- params[4]
  betas <- params[1:3]
  lin_pred <- X %*% betas
  lik <- sum(dgamma(y, shape = alph, rate = alph * lin_pred, log = T))

  # Priors
  prior_B <- dcauchy(betas, 0, c(10, 2.5, 2.5), log = T) # Gelman 2008
  prior_alph <- log(dhcauchy(alph, 20))

  # Posterior
  loglik <- sum(lik, prior_B, prior_alph)

  return(loglik)
}

# MH sampler
MH_adap <- function(y, X,
                    iters = 7500,
                    nchains = 4,
                    burn.in = 0.25,
                    acc = NULL,
                    dat,
                    start) {

  # Small check
  if(!is.null(acc) & is.null(burn.in)) {
    stop("Cannot have adaptation without burn-in!")
  }

  # Use Weighted LM and find covmat of proposal
  lm1 <- lm(1/y ~ x1 + x2, weights = y^2, data = dat,
            na.action = na.exclude)

  if(is.null(acc)) {
    var0 <- 1.7328 * vcov(lm1) # 2.38^2/3
  } else{
    var0 <- vcov(lm1)
  }

  # Storing beta samples
  post_samples <- array(NA, dim = c(iters + 1, nchains, ncol(X) + 1))
  accept <- array(NA, dim = c(iters + 1, nchains))

  # For adaptation
  if(!is.null(burn.in)){
    burn <- floor(burn.in * nrow(post_samples)) + 1
    adapt_index <- seq(601, burn, by = 100)
    # Give it 500 initial iterations without adapting
  }

  # Initialize chains
  for(j in 1:nchains){
```

```
var <- var0 # explicitly set covmat for each chain

# Initialize iterations
for (i in 1:iters){

  # Starting values
  if (any((X %*% lm1$coefficients) <= 0)) {
    betas_init <- init(start, var, X)
    post_samples[1,j,] <- c(betas_init, runif(1, 0.5, 1.5))
  } else {
    post_samples[1,j,] <- c(lm1$coefficients, runif(1, 0.5, 1.5))
  }



  # Proposal
  x_tmin1 <- post_samples[i,j,]
  prop <- proposal(x_tmin1, var, X)

  # Correction factor
  num_c <- dlnorm(x_tmin1[4], log(prop[4]), 0.1, log = T)
  denom_c <- dlnorm(prop[4], log(x_tmin1[4]), 0.1, log = T)

  # Likelihood check
  ratio <- - prop[2] / prop[3]
  linpred <- X %*% prop[1:3]

  # This is the equivalent of rejection sampling!
  if(any(linpred <= 0) | ratio <= 0 | prop[2] <= 0) {
    prob <- 0  # Set to zero if conditions violated
  } else {
    proposal_prob <- post(prop, X, y) + num_c
    current_prob <- post(x_tmin1, X, y) + denom_c
    prob <- min(1, exp(proposal_prob - current_prob))
  }
```

```
    # Accept or reject
    if (runif(1) < prob){
      post_samples[i+1,j,] <- prop # Accept
      accept[i,j] <- 1
    } else {
      post_samples[i+1,j,] <- x_tmin1 # Reject
      accept[i,j] <- 0
    }

    # Adaptation every 100 iterations - only during burn-in
    if(!is.null(acc)) {
      if(i %in% adapt_index) {
        curr_adapt <- sum(accept[(i - 100):i,j]) /
          length(accept[(i - 100):i,j])
        factor <- curr_adapt / acc
        var <- var * factor
      }
    }
  }
}

accept <- accept[-nrow(accept),]

if(!is.null(burn.in)) {
  final_samps <- post_samples[burn:nrow(post_samples),,]
  accept <- accept[burn:nrow(accept),]
  accept <- colSums(accept) / nrow(accept)
  return(list(post = final_samps, accept = accept))

} else {
  accept <- colSums(accept) / nrow(accept)
  return(list(post = post_samples, accept = accept))
}
```

```
}
```

The adaptation occurs during the burn-in phase, where the variance-covariance matrix of multivariate normal proposal Σ is scaled so as to tune the acceptance rate to a desired number (for example 44%, specified by setting `acc = 0.44`). We can visualise what this scaling implies for the parameter sampling space in Figure B.1 below.

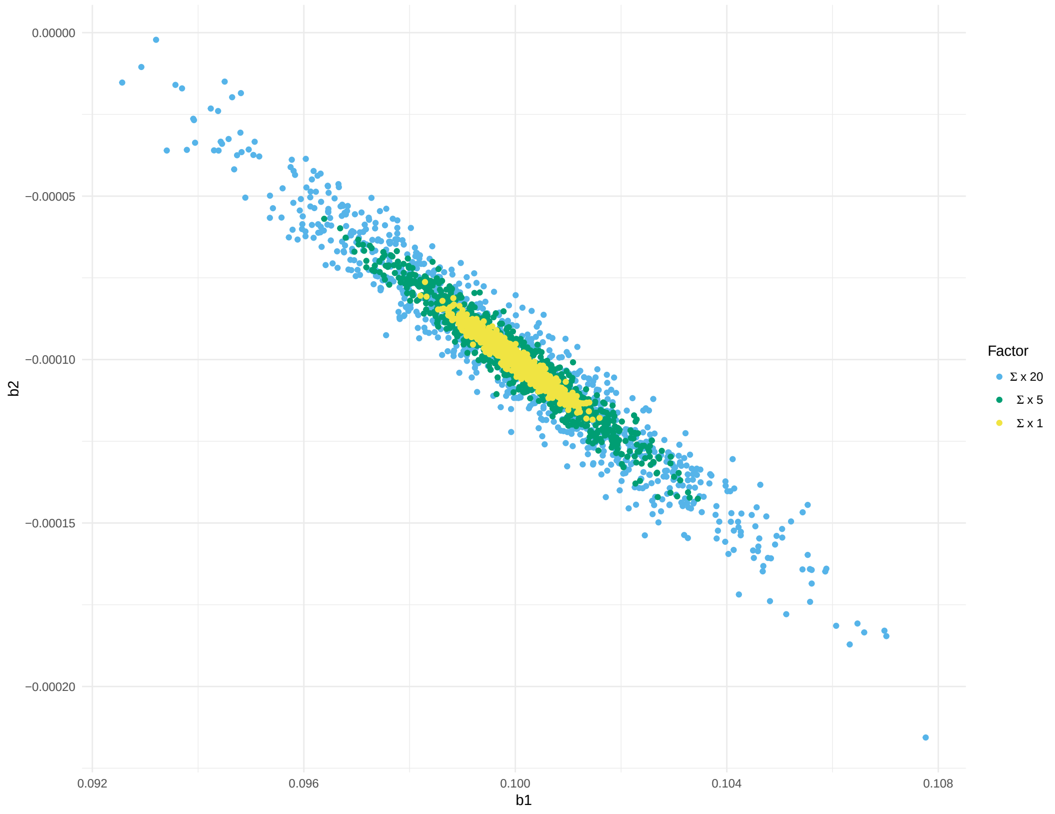

Figure B.1: Visualising the effect of scaling the variance-covariance matrix of the proposal Σ on the parameter space between β_1 and β_2.

B.4.3 MH with simulated annealing

The below sampler uses simulated annealing to find the MAP estimates.

```
# Proposal
```

```
propos_anneal <- function(params, var, X) {
  propos <- mvrnorm(1, mu = params[1:3], Sigma = var)
  alpha <- rlnorm(1, mean = log(params[4]), sd = 0.1)
  return(c(propos, alpha))
}

# Log-posterior
post_anneal <- function(params, X, y) {

  # Likelihood
  alph <- params[4]
  betas <- params[1:3]
  lin_pred <- X %*% betas
  lik <- sum(dgamma(y, shape = alph, rate = alph * lin_pred, log = T))

  # Priors
  prior_B <- dcauchy(betas, 0, c(10, 2.5, 2.5), log = T) # Gelman 2008
  prior_alph <- log(dhcauchy(alph, 20))

  # Posterior
  loglik <- sum(lik, prior_B, prior_alph)

  return(loglik)
}

# MH sampler
MH_annealing <- function(y, X, iters = 7500,
                         nchains = 4,
                         burn.in = 0.25,
                         acc = NULL,
                         dat) {

  # Small check
  if(!is.null(acc) & is.null(burn.in)) {
```

```
  stop("Cannot have adaptation without burn-in!")
}

# Use weighted LM and find covmat of proposal
lm1 <- glm(1/y ~ x1 + x2, weights = y^2, data = dat,
           family = gaussian())

if(is.null(acc)) {
  var <- 1.7328 * vcov(lm1) # 2.38^2/3
} else{
  var <- vcov(lm1)
}

# Storing beta samples
post_samples <- array(NA, dim = c(iters + 1, nchains, ncol(X) + 1))
accept <- array(NA, dim = c(iters + 1, nchains))

# For adaptation
if(!is.null(burn.in)) {
  burn <- floor(burn.in * nrow(post_samples)) + 1
  adapt_index <- seq(601, burn, by = 100)
  # Give it 500 initial iterations without adapting
}

B0 <- 1 # Basline tempering value

# Initialize chains
for(j in 1:nchains) {
  B <- 1
  # Initialize iterations
  for (i in 1:iters) {

    # Starting values
    post_samples[1,j,] <- c(lm1$coefficients, runif(1, 0.5, 1.5))
```

```
# Proposal
x_tmin1 <- post_samples[i,j,]
prop <- propos_anneal(x_tmin1, var, X)

# Correction factor
num_c <- dlnorm(x_tmin1[4], log(prop[4]), 0.1)
denom_c <- dlnorm(prop[4], log(x_tmin1[4]), 0.1)

# Likelihood check
ratio <- - prop[2] / prop[3]
linpred <- X %*% prop[1:3]

if(i > burn) {
  if(any(linpred <= 0) | prop[2] < 0 | prop[3] > 0) {
    prob <- 0
  } else {
    proposal_prob <- post_anneal(prop, X, y)
    current_prob <- post_anneal(x_tmin1, X, y)

    # Here is where annealing occurs!
    prob <- min(1, exp(proposal_prob - current_prob)^B *
                  (num_c / denom_c))
  }
  B <- 1.002 * B # geometric tempering constant
} else {
  if(any(linpred <= 0) | ratio <= 0) {
    prob <- 0
  } else {
    proposal_prob <- post_anneal(prop, X, y)
    current_prob <- post_anneal(x_tmin1, X, y)
    prob <- min(1, exp(proposal_prob - current_prob) *
                  (num_c / denom_c))
  }
```

```
    }

    # Accept or reject
    if (runif(1) < prob) {
      post_samples[i+1,j,] <- prop # Accept
      accept[i,j] <- 1
    } else {
      post_samples[i+1,j,] <- x_tmin1 # Reject
      accept[i,j] <- 0
    }

    # Adaptation every 100 iterations - only during burn-in
    if(!is.null(acc)) {
      if(i %in% adapt_index) {
        curr_adapt <- sum(accept[(i - 100):i,j]) /
          length(accept[(i - 100):i,j])
        factor <- curr_adapt / acc
        var <- var * factor
      }
    }
  }
}

accept <- accept[-nrow(accept),]

if(!is.null(burn.in)) {
  final_samps <- post_samples[burn:nrow(post_samples),,]
  accept <- accept[burn:nrow(accept),]
  accept <- colSums(accept) / nrow(accept)
  return(list(post = final_samps, accept = accept))

} else {
  accept <- colSums(accept) / nrow(accept)
  return(list(post = post_samples, accept = accept))
  }
}
```

B.5 Forecasting and PIT functions

B.5.1 SI and SIR-type epidemics

Models for both SI and SIR-type use cumulative counts to forecast, hence they are combined into one function.

```
### Forecaster for SI and SIR simulated data ~

forecaster_SI_SIR <- function(data, # data on Bayesian GLM was run
                              org_data, # original (full) sim data
                              full_post, # full posterior
                              Ct_for, # C to forecast until
                              n_for, # how many forecasts to make?
                              seed, # for reproducibility
                              plot_yn, # Plot yes or no?
                              type, # pick epidemic type for plot
                              t_for, # how far to visualise plot
                              y_top) { # graphical parameter y axis

  # Set a seed
  set.seed(seed)

  # Current C_t and time
  Ct <- data[nrow(data), ]$x1
  t <- data[nrow(data), ]$time

  # Function for one set of coeffcients
  oneset <- function(post, Ct_for) {
```

```
# Call parameters
beta <- post[1:3]
alph <- post[4]

# Setting epidemic end
if(!(is.null(Ct_for))) {
  seq_for <- seq(Ct, Ct + Ct_for)
} else {
  Ct_for <- floor(max(quad(post[1], post[2], post[3])))
  if(is.na(Ct_for)) {
    Ct_for <- 1000
    seq_for <- seq(Ct, Ct + Ct_for)
  } else {
    seq_for <- seq(Ct, Ct + Ct_for)
  }
}

# Linear predictor for all Ct
y <- sapply(seq_for, function(x) {
  eta <- beta %*% c(1, x, x^2)
  if(eta <= 0) {
    return(NA)
  } else {
    tau <- rgamma(1, shape = alph, rate = alph * eta)
    return(tau)
  }
})

# Make data fram of Ct and time
dat_for <- cbind.data.frame(x1 = seq_for, y) %>%
  mutate(time = t + cumsum(y)) %>%
  select(x1, time)

return(dat_for)
```

```
}

# Make many forecasts
fors <- apply(full_post[sample(1:nrow(full_post), size =  n_for), ],
              1, function(post) oneset(post, Ct_for))

# Data frame of forecasts
dat_fors <- bind_rows(fors, .id = "id") %>%
  subset(complete.cases(.)) %>%
  as.data.frame(.)

# Make plot if desired
if(plot_yn == TRUE) {
  if(type == "SI") {
    org_data$Transition <- 1
  }

  plot_all <- ggplot(data, aes(time, x1)) + geom_point() +
    geom_point(data = dat_fors,
               aes(time, x1, col = id), alpha = 0.25) +
    theme(legend.position = "none") +
    ylim(c(0, y_top)) + xlim(c(0, t_for)) +
    geom_point(data = org_data,
               aes(time, cumsum(Transition)),
               col = "black")

  # Make a list
  list_ret <- list(big_plot = plot_all,
                   forc_data = as.data.frame(dat_fors))

  return(list_ret)

} else {
  return(as.data.frame(dat_fors))
  }
}

#*****************************************************************#

### SI and SIR PIT function ~

PIT_SI_SIR <- function(n_dats, # number of sim datasets
                       t_cut, # when do we start forecasting from
                       t_for, # this is x in t + x
                       n_for, # n forcasts per data
                       type, # SI or SIR?
                       seed) { # replicability

  # Set a seed
  set.seed(seed)

  # Simulate the data sets (stored in a list)
  dats <- replicate(n_dats, expr = {

    if(type == "SI") {
      data <- gillespie.SI(t_origin = "2019-01-01",
                           B = 0.01,
                           N = 250,
                           I_start = 1,
                           alpha = 1)
    } else {
      data <- gillespie.SIR(t_origin = "2019-01-01",
                            t_end = 1825,
                            B = 0.05,
                            Sig = 1 / 250,
                            N = 250,
                            S_start = 249,
                            I_start = 1,
                            alpha = 1) %>%
        subset(Transition == 1) %>%
        mutate(y = inter_event(.$time),
               y = ifelse(y == y[1], time[1], y), # correct first value
               x1 = cumsum(Transition),
               x2 = x1^2)
    }

    # Check if any interevent times are zero
    data$y <- ifelse(data$y == 0, 0.5, data$y)

    return(data)
  }, simplify = F)

  test <- lapply(dats, function(data) {

    # Keep only data set with C > 25, or else too little info
    test_data <- data %>%
      subset(time <= t_cut)

    if(nrow(test_data) < 25) {
      return(NA)
    } else {

      # Run Bayesian Gamma GLM
      posterior <- MH_adap(y = test_data$y,
                           X = cbind("Intercept" = 1,
                                     x1 = test_data$x1 ,
                                     x2 = test_data$x2),
                           iters = 4000,
                           nchains = 3,
                           burn.in = 0.25,
                           acc = NULL,
                           dat = test_data,
                           start = c(0, 0.1, -0.0002))

      dens <- full_dens(posterior$post)

      # Forecast!
      forecasts <- forecaster_SI_SIR(data = test_data,
                                     org_data = data,
                                     full_post = dens,
                                     Ct_for = NULL,
                                     n_for = n_for,
                                     plot_yn = FALSE,
                                     seed = 1984)

      # Get PIT values for all horizons
      pits <- sapply(t_for, function(horiz) {

        # Actual value at t + horiz
        I <- data %>%
          subset(time <= t_cut + horiz) %>%
          select(x1) %>%
          filter(row_number() == n()) %>%
          as.numeric(.)

        # Predictive distribution
        preds <- forecasts %>%
          group_by(id) %>%
          subset(time <= t_cut + horiz) %>%
          filter(row_number() == n()) %>%
          ungroup() %>%
          select(x1) %>%
          as.data.frame(.)

        PIT <- ecdf(preds[, 1])(I) # using ECDF
        return(PIT)
      })
      return(pits)
    }
  })

  # Format results
  test <- t(bind_cols(test[!is.na(test)])) %>%
    as.data.frame(.)

  colnames(test) <- as.character(t_for)

  return(test)
}
```

B.5.2 SIS

```
### Forecaster for SIS data ~

forecaster_SIS <- function(data, # data on Bayesian GLM was run
                           org_data, # original (full) data
                           full_post, # full posterior
                           sig, # recovery rate
                           n_for, # how many forecasts to make?
                           seed, # for reproducibility
                           plot_yn, # Plot yes or no?
                           t_for, # how far to visualise plot
                           y_top) { # graphical parameter y axis

  # Set a seed
  set.seed(seed)

  # Current C_t and time
  I_t <- data[nrow(data), ]$x1
```

```
t <- data[nrow(data), ]$time

# Function for one set of coeffcients
oneset <- function(post) {

  # Call parameters
  b1 <- post[2]
  alph <- post[4] / 2 # two possible events
  N <- floor(-b1 / post[3])

  dat_for <- gillespie(c = nrow(data),
                       t_origin = "2018-07-28", # last men W,
                       n_steps = 10000, # Default 5000
                       t_end = 870, # 5y (in days) is 1825
                       B = b1,
                       Sig = sig,
                       N = N,
                       I_start = I_t,
                       alpha = alph) %>%
    mutate(time = t + cumsum(y)) %>%
    select(x1, time, Transition, Date)

  return(dat_for)
}

# Make many forecasts
fors <- apply(full_post[sample(1:nrow(full_post), size = n_for), ],
              1, function(post) oneset(post))

# Data frame of forecasts
dat_fors <- bind_rows(fors, .id = "id") %>%
  subset(complete.cases(.)) %>%
  as.data.frame(.)
```

```r
  # Make plot if desired
  if(plot_yn == TRUE) {

    plot_all <- ggplot(data, aes(time, x1)) + geom_point() +
      geom_line(data = dat_fors,
                aes(time, x1, col = id), alpha = 0.5) +
      theme(legend.position = "none") +
      ylim(c(0, y_top)) + xlim(c(0, t_for)) +
      geom_point(data = org_data,
                 aes(time, x1),
                 col = "black")

    # Make a list
    list_ret <- list(big_plot = plot_all,
                     forc_data = as.data.frame(dat_fors))

    return(list_ret)

  } else {
    return(as.data.frame(dat_fors))
  }
}

#**********************************************************************#

PIT_SIS <- function(n_dats, # number of sim datasets
                    t_cut, # when do we start forecasting from
                    t_for, # this is x in t + x
                    n_for, # n forcasts per data
                    sig = 1 / 365,
                    seed) {

  # Set a seed
  set.seed(seed)
```

```r
# Simulate the data sets (stored in a list)
dats <- replicate(n_dats, expr = {

  data <- gillespie(c = 0,
                    t_origin = "2019-01-01",
                    n_steps = 7500, # Default 5000
                    t_end = 1825, # 5y (in days) is 1825
                    B = 0.02,
                    Sig = sig,
                    N = 250,
                    I_start = 1,
                    alpha = 1) %>%
    subset(Transition == 1) %>%
    mutate(y = inter_event(.$time),
           y = ifelse(y == y[1], time[1], y))

  # Check if any interevent times are zero
  data$y <- ifelse(data$y == 0, 0.5, data$y)

  return(data)
}, simplify = F)

test <- lapply(dats, function(data) {

  # Keep only data set with C > 25, or else too little info
  test_data <- data %>%
    subset(time <= t_cut)

  if(nrow(test_data) < 25) {
    return(NA)
  } else {

    # Run Bayesian Gamma GLM
```

```
posterior <- MH_adap(y = test_data$y,
                     X = cbind("Intercept" = 1,
                               x1 = test_data$x1 ,
                               x2 = test_data$x2),
                     iters = 4000,
                     nchains = 3,
                     burn.in = 0.25,
                     acc = NULL,
                     dat = test_data,
                     start = c(0, 0.1, -0.0002))

dens <- full_dens(posterior$post)

# Wrap a try catch under in case any parameters fail
forecasts <- tryCatch(expr = {
  forecaster_SIS(data = test_data,
                 org_data = data,
                 full_post = dens,
                 sig = sig,
                 n_for = 100,
                 seed = 1984,
                 plot_yn = FALSE)
}, error = function(cond) {
  return(NA)
}, warning = function(cond) {
  return(NA)
})

# Evaluate trycatch()
if(anyNA(forecasts)) {
  return(NA)
}

pits <- sapply(t_for, function(horiz) {
        # Actual value at t + horiz
        I <- data %>%
          subset(time <= t_cut + horiz) %>%
          select(x1) %>%
          filter(row_number() == n()) %>%
          as.numeric(.)

        # Predictive distribution
        preds <- forecasts %>%
          group_by(id) %>%
          subset(time <= t_cut + horiz) %>%
          filter(row_number() == n()) %>%
          ungroup() %>%
          select(x1) %>%
          as.data.frame(.)

        PIT <- ecdf(preds[, 1])(I)
        return(PIT)
      })
      return(pits)
    }
  })

  # Format results
  test <- t(bind_cols(test[!is.na(test)])) %>%
    as.data.frame(.)

  colnames(test) <- as.character(t_for)

  return(test)
}
```

B.6 Chapter-specific

B.6.1 Chapters 1-5

```
### Load Packages & Preliminary functions ~

# Clear environment
rm(list = ls())

# Necessary packages
packages <- c("foreign", # Reads SPSS files
              "MASS", # Simulates from MVN & for Gamma shape
              "survival", # To create risk-set approx.
              "deSolve", # different. eq. solver

              # Plotting and data wrangling
              "plyr", "tidyverse", "ggpubr",

              # Easy manipulation of time/date data
              "lubridate", "anytime",

              # Bayesian diagnostics and visualisation
              "ggmcmc", "coda")

# Load all
suppressWarnings(
  for(package in packages) {
    if(!require(package, character.only = T, quietly = T)) {
      install.packages(package)
      library(package, character.only = T)
    }
  }
)
```

```
rm(package, packages)

# Load support functions
source("Support Functions/UsefulFunctions.R")
source("Support Functions/DataSimulation.R")
source("Samplers/MH_Adaptive_SimAnnealing.R")
source("Samplers/MH_Adaptive.R")

# Set overall ggplot2 theme & define colour blind palette
theme_set(theme_minimal(base_size = 16))

cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

#**********************************************************************#

### Chapter 2 ~

# SI
set.seed(1984)

# change to N = 1000, B = 0.01
dat_sim_SI <- gillespie.SI(t_origin = "2019-01-01",
                           B = 0.1,
                           N = 200,
                           I_start = 1,
                           alpha = 1)

# Plot characteristic SI curve
dat_sim_SI %>%
  gather(key = Class, value = num, S, I) %>%
  ggplot(., aes(time, num, col = Class)) + geom_point(size = 2) +
  theme(legend.position="right") +
  scale_colour_manual(values = cbPalette,
                     name = "Class",
                     breaks = c("S", "I"),
                     labels = c("S", "I")) +
  xlab("Time since I = 1 (days)") +
  ylab("Number in class")

# SIS
set.seed(1984)

dat_sim_SIS <- gillespie(c = 1,
                         t_origin = "2019-01-01",
                         n_steps = 500,
                         t_end = 750,
                         B = 0.1,
                         Sig = 1 / 200,
                         N = 200,
                         I_start = 1,
                         alpha = 1)

# Figure
dat_sim_SIS %>%
  gather(key = Class, value = num, S, I) %>%
  ggplot(., aes(time, num, col = Class)) + geom_point(size = 2) +
  theme(legend.position="right") +
  scale_colour_manual(values = cbPalette,
                     name = "Class",
                     breaks = c("S", "I"),
                     labels = c("S", "I")) +
  xlab("Time since I = 1 (days)") +
  ylab("Number in class")

# SIR
set.seed(1984)
```

```r
dat_sim_SIR <- gillespie.SIR(t_origin = "2019-01-01",
                             t_end = 325,
                             B = 0.1,
                             Sig = 1 / 50,
                             N = 200,
                             S_start = 199,
                             I_start = 1,
                             alpha = 1)

dat_sim_SIR %>%
  gather(key = Class, value = num, S, I, R) %>%
  ggplot(., aes(time, num, col = Class)) + geom_point(size = 2) +
  theme(legend.position="right") +
  scale_colour_manual(values = cbPalette,
                      name = "Class",
                      breaks = c("S", "I", "R"),
                      labels = c("S", "I", "R")) +
  xlab("Time since I = 1 (days)") +
  ylab("Number in class")

#***********************************************************************#

### Chapter 3 - Gamma GLMs ~

# Visualise some different Gamma distributions
x <- seq(0.01, 5, by = 0.01)

# Same rate, different shapes
dat_gam <- cbind.data.frame(x,
                            gam_0.75_1 = dgamma(x, 0.75, 1),
                            gam_1_1 = dgamma(x, 1, 1),
                            gam_2_1 = dgamma(x, 2, 1))

gam_shapes <- dat_gam %>%
  gather(key = "dist", value = "Density", 2:4) %>%
  ggplot(., aes(x, Density, colour = dist)) +
  geom_line(size = 2, alpha = 0.75) +
  ylim(c(0, 2)) +
  scale_colour_manual(values = cbPalette,
                      name = "Distribution",
                      labels = c("Alpha = 0.75",
                                 "Alpha = 1",
                                 "Alpha = 2"))

# Exponential, different rates
dat_gam_2 <- cbind.data.frame(x,
                            gam_1_0.75 = dgamma(x, 1, 0.5),
                            gam_1_1 = dgamma(x, 1, 1),
                            gam_1_2 = dgamma(x, 1, 2))

exp_rates <- dat_gam_2 %>%
  gather(key = "dist", value = "Density", 2:4) %>%
  ggplot(., aes(x, Density, colour = dist)) +
  geom_line(size = 2, alpha = 0.75) +
  ylim(c(0, 2)) +
  scale_colour_manual(values = cbPalette,
                      name = "Distribution",
                      labels = c("Beta = 0.5",
                                 "Beta = 1",
                                 "Beta = 2"))

ggarrange(gam_shapes, exp_rates, ncol = 1, nrow = 2,
          labels = c("A", "B"))

# Simulated annealing plot example

set.seed(1984)
```

```
posterior_SI <- MH_annealing(y = dat_sim_SI$y,
                          X = cbind("Intercept" = 1,
                                    x1 = dat_sim_SI$x1 ,
                                    x2 = dat_sim_SI$x2),
                          iters = 7500, # betas0
                          nchains = 4,
                          burn.in = 0.25,
                          acc = NULL, # desired acceptance rate
                          dat = dat_sim_SI)

# Traceplot annealed
trace_anneal <- as.data.frame(posterior_SI$post[, , 2]) %>%
  mutate(iter = 1:nrow(.)) %>%
  gather(key = chain, value = coef, 1:4) %>%
  ggplot(., aes(iter, coef, colour = chain)) +
  geom_line(size = 1, alpha = 0.75) +
  scale_colour_manual(values = cbPalette,
                      name = "Chain Number",
                      labels = c("1", "2", "3", "4")) +
  xlab("Iterations") + ylab("Coefficient value") +
  theme(legend.position = "top")

# Histogram annealed
hist_anneal <- as.data.frame(posterior_SI$post[, , 2]) %>%
  gather(key = chain, value = coef, 1:4) %>%
  ggplot(., aes(x = coef)) +
  geom_histogram(bins = 40,
                 col = cbPalette[4],
                 fill = cbPalette[1],
                 alpha = 0.75) +
  xlab("Coefficient value") + ylab("Count") +
  xlim(c(0.09, 0.12))

ggarrange(trace_anneal, hist_anneal, ncol = 1, nrow = 2,
          labels = c("A", "B"))

#**********************************************************************#

### Chapter 4 ~

# Interevent vs I plots
# for SIS and SIR make sure to only subset trans 1
intev_SI <- ggplot(dat_sim_SI, aes(x1, y)) +
  geom_point(colour = cbPalette[1], alpha = 0.75) +
  xlab("# Infectious") +
  ylab("Interevent time (days)")

intev_SIS <- dat_sim_SIS %>%
  subset(Transition == 1) %>%
  mutate(y = inter_event(.$time),
         y = ifelse(y == y[1], time[1], y)) %>%

  ggplot(., aes(x1, y)) +
  geom_point(colour = cbPalette[1], alpha = 0.75) +
  xlab("# Infectious") +
  ylab("Interevent time (days)")

intev_SIR <- dat_sim_SIR %>%
  subset(Transition == 1) %>%
  mutate(y = inter_event(.$time),
         y = ifelse(y == y[1], time[1], y),
         x1 = cumsum(Transition),
         x2 = x1^2) %>%
  ggplot(., aes(x1, y)) +
  geom_point(colour = cbPalette[1], alpha = 0.75) +
  xlab("# Infectious") +
  ylab("Interevent time (days)")
```

```
# Interevs figure
ggarrange(intev_SI, intev_SIS, intev_SIR, ncol = 1, nrow = 3,
          labels = c("A", "B", "C"), hjust = -73)

#**********************************************************************#

### Chapter 5 ~

# Show three different forecasts SI

# No forecasts
no_forc <- dat_sim_SI[1:100, ] %>%
  ggplot(., aes(time, I)) +
  geom_point(size = 2, col = cbPalette[1]) +
  xlab("Time since I = 1 (days)") +
  ylab("# Infectious") +
  xlim(c(0, 100)) + ylim(c(0, 250))

set.seed(1984)

three_forc <- no_forc +
  # B = 0.12, N = 230, alpha = 1.1
  geom_point(data = gillespie.SI(t_origin = dat_sim_SI[100, ]$Date,
                                 B = 0.12,
                                 N = 230,
                                 I_start = dat_sim_SI[100, ]$I,
                                 alpha = 1.1),
             aes(time + dat_sim_SI[100, ]$time, I),
             col = cbPalette[2]) +

  # B = 0.09, N = 185, alpha = 1.05
  geom_point(data = gillespie.SI(t_origin = dat_sim_SI[100, ]$Date,
                                 B = 0.09,
                                 N = 185,
                                 I_start = dat_sim_SI[100, ]$I,
                                 alpha = 1.05),
             aes(time + dat_sim_SI[100, ]$time, I),
             col = cbPalette[3]) +

  # B = 0.10, N = 205, alpha = 0.9
  geom_point(data = gillespie.SI(t_origin = dat_sim_SI[100, ]$Date,
                                 B = 0.10,
                                 N = 205,
                                 I_start = dat_sim_SI[100, ]$I,
                                 alpha = 0.9),
             aes(time + dat_sim_SI[100, ]$time, I),
             col = cbPalette[4])

ggarrange(no_forc, three_forc, ncol = 2, nrow = 1,
          labels = c("A", "B"))

# Show what an inverse polynomial function looks like

inv_poly <- Vectorize(function(x, b0 = 0, b1 = 0.1, b2 = -0.0005) {
  mu <- (b0 + b1 * x + b2 * x^2)^-1
  return(mu)
})

inv_poly_plot <- cbind.data.frame(x = 1:250,
                 f = inv_poly(x = 1:250)) %>%
  ggplot(., aes(x, f)) +
  geom_point(size = 2, colour = cbPalette[1]) +
  ylab("Predicted interevent time (days)") +
  xlab("Cumulative infections (C)")

inv_cumul <- cbind.data.frame(time = cumsum(inv_poly(x = 1:199)),
                              C = 1:199) %>%
  ggplot(., aes(time, C)) +
  geom_point(size = 2, col = cbPalette[1]) +
  ylab("Cumulative infections (C)") +
  xlab("Time since C = 1 (days)")

ggarrange(inv_poly_plot, inv_cumul, nrow = 1, ncol = 2,
          labels = c("A", "B"))
```

B.6.2 Chapter 6 - PIT histograms

```
### Plot deterministic cutoffs for PIT histograms ~

# Set common parameters for all
N <- 250
S_init <- 249
I_init <- 1

# For SI
time <- seq(0, 1000, by = 1) # time in days

determ_func_SI(c(S = 1 - I_init / N, I = I_init / N),
                time,
                c(beta = 0.01)) %>%
  mutate(x1 = I * N) %>%
  ggplot(., aes(time, x1)) +
  geom_line(size = 2, col = cbPalette[1]) +
  # 14 / 17 / 20 months cutoffs
  geom_vline(xintercept = c(427, 518.5, 610),
             linetype = "dashed") +
  xlab("Time since I = 1 (days)") +
  ylab("Cumulative infections (C)")

# For SIS
time <- seq(0, 1500, by = 1) # time in days
```

```r
determ_func_SIS(c(S = 1 - I_init / N, I = I_init / N),
                time,
                c(beta = 0.01, gamma = 1/366)) %>% # 0.05, 1/25
  mutate(x1 = I * N) %>%
  ggplot(., aes(time, x1)) +
  geom_line(size = 2, col = cbPalette[1]) +
  xlab("Time since I = 1 (days)") +
  ylab("# Infectious") +

  # 8, 11, 14 months
  geom_vline(xintercept = c(244, 335.5, 427),
             linetype = "dashed")

# For SIR
time <- seq(0, 750, by = 1) # time in days

determ_func_SIR(c(S = 1 - I_init / N, I = I_init / N, R = 0),
                time,
                c(beta = 0.05, gamma = 1 / 250)) %>%
  mutate(CI = N * (cumsum(I) / sum(I))) %>%
  # gather(key = state, value = value, CI, I) %>%
  ggplot(., aes(time, CI)) +
  geom_line(size = 2, col = cbPalette[1]) +

  # Add reference line
  geom_vline(xintercept = c(183, 274.5, 366),
              linetype = "dashed") +
  xlab("Time since I = 1 (days)") +
  ylab("Cumulative infections (C)")

#**********************************************************************#

### SI PITS ~
```

```
si_pit_cut427 <- PIT_SI_SIR(n_dats = 1000,
                            t_cut = 427,
                            t_for = c(30.5, 183, 366), # 1mo, 6mo, 1Y
                            type = "SI",
                            n_for = 100,
                            seed = 1984)

names(si_pit_cut427) <- c("fac", names(si_pit_cut518))

si_pit_cut518 <- PIT_SI_SIR(n_dats = 1000,
                            t_cut = 518.5,
                            t_for = c(30.5, 183, 366), # 1mo, 6mo, 1Y
                            type = "SI",
                            n_for = 100,
                            seed = 1984)

si_pit_cut610 <- PIT_SI_SIR(n_dats = 1000,
                            t_cut = 610,
                            t_for = c(30.5, 183, 366), # 1mo, 6mo, 1Y
                            type = "SI",
                            n_for = 100,
                            seed = 1984)

# Use final .csv file to plot
dat_pit_SI <- read.csv("dat_pit_SI.csv") %>%
  mutate(cutoff = factor(cutoff, levels = c("14 months",
                                            "17 months",
                                            "20 months"))) %>%
  mutate(horizon = factor(horizon, levels = c(30.5, 183, 366)))

p_SI <- ggplot(dat_pit_SI, aes(PIT)) +
  geom_histogram(aes(y=..density..),
                 bins = 15,
                 col = "black",
                 fill = cbPalette[1],
                 alpha = 0.75) +
  geom_hline(yintercept = 1, linetype = "dotted") +
  ylab("Relative frequency")

facet(p_SI, facet.by = c("horizon", "cutoff"),
      panel.labs.background = list(color = "darkblue",
                                   fill = "darkblue",
                                   size = 0.5),
      panel.labs.font = list(color = "white"),
      panel.labs.font.x = list(color = "white"),
      panel.labs = list(cutoff = c("Cutoff: 14 mos.",
                                   "Cutoff: 17  mos.",
                                   "Cutoff: 20 mos."),
                        horizon = c("1 mo. ahead",
                                    "6 mos. ahead",
                                    "1 yr. ahead")))

#*******************************************************************#

### PIT for SIR (cutoffs = (183, 274.5, 366 days) ~

sir_pit_cut183 <- PIT_SI_SIR(n_dats = 1000,
                             t_cut = 183,
                             t_for = c(30.5, 183, 366), # 1mo, 6mo, 1Y
                             type = "SIR",
                             n_for = 100,
                             seed = 1984)

sir_pit_cut274 <- PIT_SI_SIR(n_dats = 1000,
                             t_cut = 274.5,
                             t_for = c(30.5, 183, 366), # 1mo, 6mo, 1Y
                             type = "SIR",
                             n_for = 100,
                             seed = 1984)

sir_pit_cut366 <- PIT_SI_SIR(n_dats = 1000,
                             t_cut = 366,
                             t_for = c(30.5, 183, 366), # 1mo, 6mo, 1Y
                             type = "SIR",
                             n_for = 100,
                             seed = 1984)

# SIR PIT
dat_pit_SIR <- read.csv("dat_pit_SIR.csv") %>%
  mutate(cutoff = factor(cutoff, levels = c("6 months",
                                            "9 months",
                                            "1 year"))) %>%
  mutate(horizon = factor(horizon, levels = c(30.5, 183, 366)))

p_SIR <- ggplot(dat_pit_SIR, aes(PIT)) +
  geom_histogram(aes(y=..density..),
                 bins = 15,
                 col = "black",
                 fill = cbPalette[1],
                 alpha = 0.75) +
  geom_hline(yintercept = 1, linetype = "dotted") +
  ylab("Relative frequency")

facet(p_SIR, facet.by = c("horizon", "cutoff"),
      panel.labs.background = list(color = "darkblue",
                                   fill = "darkblue",
                                   size = 0.5),
      panel.labs.font = list(color = "white"),
      panel.labs.font.x = list(color = "white"),
      panel.labs = list(cutoff = c("Cutoff: 6 mos.",
                                   "Cutoff: 9  mos.",
                                 "Cutoff: 1 yr."),
                        horizon = c("1 mo. ahead",
                                    "6 mos. ahead",
                                    "1 yr. ahead")))

#**********************************************************************#

### PIT for SIS ~

sis_pit_cut244 <- PIT_SIS(n_dats = 1000,
                          t_cut = 244,
                          t_for = c(30.5, 183, 366),
                          sig = 1 / 365,
                          n_for = 100,
                          seed = 1984)

sis_pit_cut335 <- PIT_SIS(n_dats = 1000,
                          t_cut = 335.5,
                          t_for = c(30.5, 183, 366),
                          sig = 1 / 365,
                          n_for = 100,
                          seed = 1984)

sis_pit_cut427 <- PIT_SIS(n_dats = 1000,
                          t_cut = 427,
                          t_for = c(30.5, 183, 366),
                          sig = 1 / 365,
                          n_for = 100,
                          seed = 1984)

dat_pit_SIS <- read.csv("dat_pit_SIS.csv") %>%
  mutate(cutoff = factor(cutoff, levels = c("8 months",
                                            "11 months",
                                            "14 months"))) %>%
  mutate(horizon = factor(horizon, levels = c(30.5, 183, 366)))

p_SIS <- ggplot(dat_pit_SIS, aes(PIT)) +
  geom_histogram(aes(y=..density..),
                 bins = 15,
                 col = "black",
                 fill = cbPalette[1],
                 alpha = 0.75) +
  geom_hline(yintercept = 1, linetype = "dotted") +
  ylab("Relative frequency")

facet(p_SIS, facet.by = c("horizon", "cutoff"),
      panel.labs.background = list(color = "darkblue",
                                   fill = "darkblue",
                                   size = 0.5),
      panel.labs.font = list(color = "white"),
      panel.labs.font.x = list(color = "white"),
      panel.labs = list(cutoff = c("Cutoff: 8 mos.",
                                   "Cutoff: 11  mos.",
                                   "Cutoff: 14 mos."),
                        horizon = c("1 mo. ahead",
                                    "6 mos. ahead",
                                    "1 yr. ahead")))
```

B.6.3 Chapter 7 - MenW analysis

```
library(xtable) # for tables later on

dat_men <- read.csv("Data/Data MenW 2012-juli 2018.csv",
                    stringsAsFactors = F)

# Data until end July 18'
dat_men <- dat_men %>%
  mutate(Date = as.Date(datum_EZD, "%d/%m/%Y"),
         time = time_since(Date)) %>%
  arrange(time) %>%
  mutate(end = time + 335, status = 1) %>%
  mutate(y = inter_event(time),
         x1 = infect(.),
         x2 = x1^2) %>%
  slice(-1) %>%
  select(-c(end, status))

# Bar plot
bar_men <- dat_men %>%
  arrange(time) %>%
  mutate(ncases = 1) %>%

  # Change between year/month
  group_by(month = floor_date(Date, "halfyear")) %>%
  summarize(Infec = sum(ncases)) %>%
  mutate(id = "1")

# Remove fill for no colours
ggplot(bar_men, aes(x = month, y = Infec)) +
  geom_bar(stat = "identity", col = "black",
           fill = cbPalette[1], alpha = 0.75) +
  xlab("Year") + ylab("# Bi-annual New Cases")

# Cuminc plot
cuminc_men <- dat_men %>%
  mutate(ncases = 1) %>%
  ggplot(., aes(x = Date, y = cumsum(ncases))) +
  geom_point(size = 2, col = cbPalette[1]) +
  ylab("Cumulative infections (C)") + xlab("Year")

risk_set_men <- dat_men %>%
  ggplot(., aes(x = Date, y = x1)) +
  geom_point(size = 2, col = cbPalette[1]) +
  ylab("Approximate # Infectious (I)") + xlab("Year")

# Export this
ggarrange(cuminc_men, risk_set_men, nrow = 1, ncol = 2,
          labels = c("A", "B"))

# Run sampler
set.seed(1984)

posterior_men <- MH_adap(y = dat_men$y,
                         X = cbind("Intercept" = 1,
                                   x1 = dat_men$x1 ,
                                   x2 = dat_men$x2),
                         iters = 10000,
                         nchains = 4,
                         burn.in = 0.25,
                         acc = NULL,
                         dat = dat_men)

# Prepare for posterior diagnostics
post_summary <- ggpost_convert(posterior_men$post)
dens <- full_dens(posterior_men$post)

# Get MAP estimates - annealed sampler
set.seed(1984)

posterior_men_anneal <- MH_annealing(y = dat_men$y,
                                     X = cbind("Intercept" = 1,
                                               x1 = dat_men$x1 ,
                                               x2 = dat_men$x2),
                                     iters = 10000,
                                     nchains = 4,
                                     burn.in = 0.25,
                                     acc = NULL,
                                     dat = dat_men)

# MAP, mean and median coefs
MAP_men <- apply(full_dens(posterior_men_anneal$post),
                 2, function(x) Mode(x))

mean_men <- apply(full_dens(posterior_men$post),
                  2, function(x) mean(x))

median_men <- apply(full_dens(posterior_men$post),
                    2, function(x) quantile(x, 0.5))

# Visualise chains and densities
trace_men <- ggs_traceplot(post_summary$ggmc) + ylab("Value") +
  aes(alpha = 0.75) +
  scale_colour_manual(values = cbPalette,
                      name = "Chain",
                      labels = c("1", "2", "3", "4"))

dens_men <- ggs_density(post_summary$ggmc) + ylab("Density") +
  aes(alpha = 0.75) +  xlab("Value") +
  scale_colour_manual(values = cbPalette,
                      name = "Chain",
                      labels = c("1", "2", "3", "4"))

ggarrange(trace_men, dens_men, ncol = 2, nrow = 1,
          common.legend = TRUE,
          labels = c("A", "B"))

# Autocorrelation plot
ggs_autocorrelation(post_summary$ggmc) +
  scale_fill_manual(name = NULL,
                    values = cbPalette[1:4],
                    labels = c("Alpha", "B0", "B1", "B2")) +
  aes(alpha = 0.75) +
  theme(legend.position = "none")

# Plot fit to data
X <- cbind("Intercept" = 1,
           x1 = dat_men$x1 ,
           x2 = dat_men$x2)

Q05 <- apply(dens, 2, function(x) quantile(x, 0.05))[1:3]
Q95 <- apply(dens, 2, function(x) quantile(x, 0.95))[1:3]
conf <- cbind.data.frame(Q05 = (X %*% Q05)^-1,
                         Q95 = (X %*% Q95)^-1)

# Nice plot with credible intervals, only MAP plotted
ggplot(dat_men, aes(x1, y)) + geom_point() +
  geom_ribbon(aes(ymin = conf$Q05, ymax = conf$Q95), alpha = 0.3) +
  geom_line(aes(y = (X %*% MAP_men[1:3])^-1),
            size = 1, colour = "blue") +
  ylim(c(0, 300)) +
  xlab("# Infectious") + ylab("Interevent time (days)")

# Median estimates + 90% CIs
apply(dens, 2, function(x) quantile(x, c(0.05, 0.5, 0.95)))

xtable(apply(dens, 2, function(x) quantile(x, c(0.05, 0.5, 0.95))),
       digits = -10)

# Diagnostics
summary(post_summary$coda_ob)

posterior_men$accept
effectiveSize(post_summary$coda_ob)
```

```
gelman.diag(post_summary$coda_ob)

# Reproductive number
R0 <- dens[, 2] * 335
c(mean(R0), Mode(full_dens(posterior_men_anneal$post)[, 2] * 335),
  quantile(R0, c(0.5, 0.05, 0.95)))

# Make histograms of N and R0
hist_R0 <- ggplot(data = as.data.frame(R0), aes(R0)) +
  geom_histogram(bins = 50,
                 col = "black",
                 fill = cbPalette[1],
                 alpha = 0.75) + ylab("Count") +
  geom_vline(xintercept = unname(quantile(R0, c(0.05, 0.5, 0.95))),
             linetype = "dashed")

hist_N <- ggplot(data = data.frame(N = dens[, 5]), aes(N)) +
  geom_histogram(bins = 50,
                 col = "black",
                 fill = cbPalette[1],
                 alpha = 0.75) + ylab("Count") +
  xlim(c(0, 1500)) +
  geom_vline(xintercept = unname(quantile(dens[, 5],
                                          c(0.05, 0.5, 0.95))),
             linetype = "dashed")

ggarrange(hist_N, hist_R0, ncol = 2, nrow = 1,
          labels = c("A", "B"))

#**********************************************************************#

# Run the frequentist glm
glm_men <- glm(y ~ x1 + x2, data = dat_men,
               family = Gamma(link = "inverse"),
               start = MAP_men[1:3])

freq_coefs <- glm_men$coefficients

# N from the GLM
N_freq <- -freq_coefs[2] / freq_coefs[3]

# R0 from GLM
R0_freq <- freq_coefs[2] * 335

# I* from GLM
N_freq * (1 - R0_freq^-1)

# Shape estimates
gamma.shape(glm_men) # MLE, using MASS package
  summary(glm_men)$dispersion^-1 # Using McCullagh method

# Get appropriate MLE summary and CIs
# summary(glm_men) for normal summary

sum_glm <- summary(glm_men, dispersion = gamma.dispersion(glm_men))

# Compute CIs based on asymptotic normality
CI_low <- freq_coefs - qnorm(0.95) * sum_glm$coefficients[, 2]
CI_high <- freq_coefs + qnorm(0.95) * sum_glm$coefficients[, 2]

# Deviance based tests
anova(glm_men, test = "Chisq") # added sequentially

# Chi square of deviance, between null and quadrativ
pchisq(glm_men$null.deviance - glm_men$deviance,
       df = 2, lower.tail = F)

# The above is the same as the below
```

```
glm_men0 <- glm(y ~ 1, data = dat_men,
               family = Gamma(link = "inverse"))

anova(glm_men0, glm_men, test = "Chisq")

# Summarise all in latex table
library(xtable)

cbind.data.frame(sum_glm$coefficients,
                 CI_low, CI_high) %>%
  select(-"z value") %>%
  xtable(., digits = -2)

#**********************************************************************#

# Forecast the MenW epidemic -
forecasts_men <- forecaster_SIS(data = dat_men,
                                org_data = dat_men,
                                full_post = dens,
                                sig = 1 / 335,
                                n_for = 100,
                                seed = 1984,
                                plot_yn = TRUE,
                                t_for = 3250,
                                y_top = 250)

# Set up deterministic forcasts for plotting
dat_deter <- forecasts_men$forc_data %>%
  subset(id == 1) %>%
  select(time)

# Last seen x1
It <- dat_men$x1[length(dat_men$x1)]
```

```
# For MAP estimates
N <- floor(MAP_men["N"])
B <- unname(MAP_men["B1"])

determ_MAP <- determ_func_SIS(c(S = 1 - It / N, I = It / N),
                     as.numeric(dat_deter$time),
                     c(beta = B, gamma = 1 / 335)) %>%
  mutate(x1 = I.N * N)

# For Median estimates
N <- floor(median_men["N"])
B <- unname(median_men["B1"])

determ_median <- determ_func_SIS(c(S = 1 - It / N, I = It / N),
                             as.numeric(dat_deter$time),
                             c(beta = B, gamma = 1 / 335)) %>%
  mutate(x1 = I.N * N)

# For mean estimates
N <- floor(mean_men["N"])
B <- unname(mean_men["B1"])

determ_mean <- determ_func_SIS(c(S = 1 - It / N, I = It / N),
                           as.numeric(dat_deter$time),
                           c(beta = B, gamma = 1 / 335)) %>%
  mutate(x1 = I.N * N)

# Plot as function of date
cloud_fors <- ggplot(dat_men, aes(Date, x1)) +
  geom_point(size = 1) +
  geom_line(data = forecasts_men$forc_data,
          aes(Date, x1, col = id), alpha = 0.25) +
  ylab("# Infectious") + xlab("Year") +
  # Add MAP, mean and median determinstic forecasts
  geom_line(data = determ_MAP, aes(Date, x1),
            size = 1, linetype = 1) +
  geom_line(data = determ_median, aes(Date, x1),
            size = 1, linetype = 2) +
  geom_line(data = determ_mean, aes(Date, x1),
            size = 1, linetype = 3) +
  theme(legend.position = "none",
        axis.text=element_text(size=12),
        axis.title=element_text(size=14))

# Bar chart with conf intervals
bars_fors <- forecasts_men$forc_data %>%
  group_by(month = floor_date(Date, "halfyear"), id) %>%
  summarise(infec = sum(Transition)) %>%
  group_by(month) %>%
  summarise(‘0%‘ = quantile(infec, probs = 0.05),
            ‘90%‘ = quantile(infec, probs = 0.95),
            Infec = median(infec)) %>%
  filter(month < "2021-01-01") %>%
  mutate(id = "0") %>%
  bind_rows(bar_men) %>%
  ggplot(., aes(month, Infec, fill = factor(id))) +
  geom_bar(stat = "identity", col = "black", alpha = 0.75) +
  geom_errorbar(aes(ymin = ‘0%‘, ymax = ‘90%‘), width = 50,
                position = "dodge") + theme_minimal() +
  ylab("# Bi-annual New Cases") + xlab("Year") +
  scale_fill_manual(values = c(cbPalette[3], cbPalette[1]),
                    name=NULL,
                    labels=c("Forecasted", "Data")) +
  theme(legend.position = "top",
        axis.text=element_text(size=12),
        axis.title=element_text(size=14))
```

```
# Bring the two together
ggarrange(bars_fors, cloud_fors, nrow = 2, ncol = 1,
          labels = c("A", "B"), common.legend = TRUE)

# Table of all estimates
tab <- rbind(mean_men,
             MAP_men,
             median_men,
             five = apply(dens, 2,
                          function(x) quantile(x, 0.05)),
             ninetyfive = apply(dens, 2,
                                function(x) quantile(x, 0.95)))

xtable(t(tab)[5:6,], digits = 2)

# Hist of interevent times
ggplot(dat_men, aes(y)) +
  geom_histogram(aes(y=..count..),
                 bins = 100,
                 col = "black",
                 fill = cbPalette[1],
                 alpha = 0.75) +
  ylab("Count") + xlab("Interevent time (days)")

#**********************************************************************#

### Plot of parameter space (for appendix) ~

lm1 <- glm(1/y ~ x1 + x2, weights = y^2, data = dat_men,
           family = gaussian())
var <- vcov(lm1)
params <- c(0, 0.1, -0.0001)

data.frame(mvrnorm(n = 1000, mu = params, Sigma = var)) %>%
  bind_rows(data.frame(mvrnorm(n = 1000, mu = params,
                               Sigma = var * 5)),
            data.frame(mvrnorm(n = 1000, mu = params,
                               Sigma = var * 20))) %>%
  mutate(factor = factor(c(rep(3, 1000),
                           rep(2, 1000),
                           rep(1, 1000)))) %>%
  arrange(factor) %>%
  ggplot(., aes(x1, x2, color = factor)) + geom_point() +
  theme_minimal()  +
  xlab("b1") +
  ylab("b2") +
  scale_color_manual(name = "Factor",
                     labels = c(expression(paste(Sigma, " x 20")),
                                expression(paste(Sigma, " x 5")),
                                expression(paste(Sigma, " x 1"))),
                     values = cbPalette[3:5])
```

B.7 Alternative samplers

B.7.1 Stan

Here we demonstrate the use of Stan and its applied regression modelling package on the MenW dataset. For a discussion about how Stan deals with the inverse link, please refer to this thread.

```
### Calling Stan via Rstan ~


library(rstan)
library(rstanarm)
library(shinystan)


options(mc.cores = parallel::detectCores())
```

```r
rstan_options(auto_write = TRUE)

# Fit model of MenW data
fit_men <- stan_glm(y ~ x1 + x2, data = dat_men,
                    family = Gamma(link="inverse"),
                    prior = cauchy(0, 2.5),
                    prior_intercept = cauchy(0, 2.5),
                    prior_aux = exponential(0.1))

summary(fit_men, probs = c(.025, .975), digits = 5)
stan_trace(fit_men)
stan_hist(fit_men)

# For fancy shinystan interface
launch_shinystan(fit_sim)

#*******************************************************************#

### Demonstration of "raw" stan code, using the log-link ~

# Simulate some data
set.seed(1984)

N <- 50
X <- cbind(1, runif(N, 0, 2), rnorm(N, 2, 1))

betas <- c(0, 2, 0.5)
y <- rexp(N, rate = 1 / X %*% betas)

stan_code <- "
data {
int<lower=0> N; //the number of observations
int<lower=0> K; //the number of columns in the model matrix
matrix[N, K] X; //the model matrix
vector[N] y; //the response
}
parameters {
vector[K] betas;
real<lower=0.001> shape;
}
model {
// Priors
betas ~ normal(0,50);
shape ~ exponential(0.1);

// Evaluate likelihood
for(i in 1:N)
// For inv. link, we would write rate = shape * (X[i, ] * betas)
y[i] ~ gamma(shape, (shape / exp(X[i, ] * betas)));
}
"

m_stan <- stan(model_code = stan_code,
               data = list(X = X,
                           y = y,
                           N = N,
                           K = ncol(X)))

stan_trace(m_stan)
```

Bibliography

Roy M Anderson and Robert M May. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

Thomas F. Bateson. Gamma regression of interevent waiting times versus poisson regression of daily event counts: Inside the epidemiologist's toolbox-selecting the best modeling tools for the job, 2009. ISSN 10443983.

Michael Carney and Pádraig Cunningham. Evaluating density forecasting models. *Department of Computer Science, Trinity College Dublin*, pages 1–12, 2006.

Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. <i>Stan</i> : A Probabilistic Programming Language. *Journal of Statistical Software*, 2017. ISSN 1548-7660. doi: 10.18637/jss.v076.i01.

William Checkley, Judith Guzman-Cottrill, Leonardo Epstein, Nancy Innocentini, Jonathan Patz, and Stanford Shulman. Short-term weather variability in chicago and hospitalizations for kawasaki disease. *Epidemiology*, 2009. ISSN 10443983. doi: 10.1097/EDE.0b013e3181961a9b.

P. G. Coen, K. Cartwright, and J. Stuart. Mathematical modelling of infection and disease due to Neisseria meningitidis and Neisseria lactamica. *International Journal of Epidemiology*, 2000. ISSN 03005771. doi: 10.1093/ije/29.1.180.

David B Dahl, David Scott, Charles Roosen, Arni Magnusson, and Jonathan Swinton. *xtable: Export Tables to LaTeX or HTML*, 2018. URL https://cran.r-project.org/package=xtable.

Philippe De Wals and André Bouckaert. Methods for estimating the duration of bacterial carriage. *International Journal of Epidemiology*, 1985. ISSN 03005771. doi: 10.1093/ije/14.4.628.

Ebola Response Team. Ebola Virus Disease in West Africa — The First 9 Months of the Epidemic and Forward Projections. *New England Journal of Medicine*, 2014. ISSN 2353883X. doi: 10.15678/EBER.2017.050110.

Dirk Eddelbuettel. *anytime: Anything to 'POSIXct' or 'Date' Converter*, 2018. URL https://cran.r-project.org/package=anytime.

Fadlalla G. Elfadaly and Paul H. Garthwaite. Eliciting prior distributions for extra parameters in some generalized linear models. *Statistical Modelling*, 2015. ISSN 14770342. doi: 10.1177/1471082X14553374.

Julian J Faraway. *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models.* 2006. ISBN 0-203-49228-5. doi: 10.1111/j.1541-0420.2006.00596_12.x.

Xavier Fernández-i Marín. ggmcmc: Analysis of MCMC Samples and Bayesian Inference. *Journal of Statistical Software*, 70(9):1–20, 2016. doi: 10.18637/jss.v070.i09.

E. C. Fieller. Some Problems in Interval Estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1954. ISSN 00359246. doi: 10.1111/j.2517-6161.1954.tb00159.x.

Thomas R Frieden, Harold W Jaffe, Chesley L Richards, Michael F Iademarco, Charlotte K Kent, Martha F Boyd, Maureen A Leahy, Julia C Martinroe, Stephen R Spriggs, Terraye M Starr, and William L Roper. Estimating the Future Number of Cases in the Ebola Epidemic — Liberia and Sierra Leone, 2014-2015. *Centers for Disease Control and Prevention Mortality and Morbidity Weekly Report*, 63(3): 1–14, 2014. ISSN 1545-8636. doi: su6303a1[pii].

Jonah Gabry. Package 'rstanarm' - Bayesian Applied Regression Modeling via Stan. *R documentation*, 2018.

Sylvain Gandon, Troy Day, C. Jessica E. Metcalf, and Bryan T. Grenfell. Forecasting Epidemiological and Evolutionary Dynamics of Infectious Diseases, 2016. ISSN 01695347.

A Gelman. Prior distribution for variance parameters in hierarchical models. *Bayesian Analysis*, 2006. ISSN 19360975. doi: 10.1214/06-BA117A.

Andrew Gelman and Donald B. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 1992. ISSN 0883-4237. doi: 10.1214/ss/1177011136.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis - 3ed.* 2004. ISBN 1570-0232. doi: 10.1186/1754-1611-9-2.

Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, and Yu Sung Su. A weakly informative default prior distribution for logistic and other regression models. *Annals of Applied Statistics*, 2008. ISSN 19326157. doi: 10.1214/08-AOAS191.

Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. In *Journal of Physical Chemistry*, 1977. ISBN 1749810441. doi: 10.1021/j100540a008.

Tilmann Gneiting, Adrian E. Raftery, Fadoua Balabdaoui, and Anton H. Westveld. Verifying probabilistic forecasts: Calibration and sharpness. *Proc. Workshop on Ensemble Forecasting*, (Val-Morin, QC, Canada), 2003. ISSN 1541-0420.

Garrett Grolemund and Hadley Wickham. Dates and Times Made Easy with lubridate. *Journal of Statistical Software*, 40(3):1–25, 2011. URL http://www.jstatsoft.org/v40/i03/.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Elements of Statistical Learning 2nd ed. *Elements*, 2009. ISSN 01727397. doi: 10.1007/978-0-387-84858-7.

W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 1970. ISSN 00063444. doi: 10.1093/biomet/57.1.97.

Leonhard Held, Sebastian Meyer, and Johannes Bracher. Probabilistic forecasting in infectious disease epidemiology: the 13th Armitage lecture. *Statistics in Medicine*, 2017. ISSN 10970258. doi: 10.1002/sim.7363.

Albert Jan Van Hoek, Mirjam Knol, Hester De Melker, and Jacco Wallinga. Optimizing the use of the Men-ACWY conjugated vaccine to control the developing meningococcal W disease outbreak in the Netherlands , a rapid analysis . *bioRxiv*, 2018.

Adam M Johansen, Ludger Evers, and Lecture Notes. Johansen, Evers, Notes - 2007 - Monte Carlo Methods. 2007.

Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 1995. ISSN 1537274X. doi: 10.1080/01621459.1995.10476572.

Alboukadel Kassambara. *ggpubr: 'ggplot2' Based Publication Ready Plots*, 2018. URL https://cran.r-project.org/package=ggpubr.

Matthew James Keeling, Pejman Rohani, and Babak Pourbohloul. Modeling Infectious Diseases in Humans and Animals:Modeling Infectious Diseases in Humans and Animals. *Clinical Infectious Diseases*, 2008. ISSN 1058-4838. doi: 10.1086/591197.

W. O. Kermack and A. G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 1927. ISSN 00928240. doi: 10.1007/BF02464423.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 1983. ISSN 00368075. doi: 10.1126/science.220.4598.671.

Mirjam J. Knol, Susan J.M. Hahné, Jay Lucidarme, Helen Campbell, Hester E. de Melker, Stephen J. Gray, Ray Borrow, Shamez N. Ladhani, Mary E. Ramsay, and Arie van der Ende. Temporal associations between national outbreaks of meningococcal serogroup W and C disease in the Netherlands and England: an observational cohort study. *The Lancet Public Health*, 2017. ISSN 24682667. doi: 10.1016/S2468-2667(17)30157-3.

Mirjam J. Knol, Wilhelmina L.M. Ruijs, Laura Antonise-Kamp, Hester E. de Melker, and Arie van der Ende. Implementation of MenACWY vaccination because of ongoing increase in serogroup W invasive meningococcal disease, the Netherlands, 2018. *Eurosurveillance*, 2018. ISSN 15607917. doi: 10.2807/1560-7917.ES.2018.23.16.18-00158.

Alun Lloyd. Introduction to Epidemiological Modeling : Basic Models and Their Properties. *Networks*, pages 1–166, 2007. ISSN 0043373X. doi: 10.2307/1500086.

Mark Landry. Machine Learning with R and H2O. Technical report, 2017.

P McCullagh and J A Nelder. *Generalized Linear Models, Second Edition*. 1989. ISBN 0412317605. doi: 10.1007/978-1-4899-3242-6.

Elaine O. Nsoesie, John S. Brownstein, Naren Ramakrishnan, and Madhav V. Marathe. A systematic review of studies on forecasting the dynamics of influenza outbreaks, 2014. ISSN 17502659.

Martyn Plummer. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. In *DSC 2003 Working Papers*, 2003. ISBN ISSN 1609-395X. doi: 10.1.1.13.3406.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. CODA: Convergence Diagnosis and Output Analysis for MCMC. *R News*, 2006. ISSN 1098-6596.

R Core Team. *foreign: Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...*, 2017. URL `https://cran.r-project.org/package=foreign`.

J.S. Rosenthal and Others. Optimal proposal distributions and adaptive MCMC. *Preprint*, 2008. ISSN 1098-6596. doi: 10.1201/b10905.

Karline Soetaert, Thomas Petzoldt, and R Woodrow Setzer. Solving Differential Equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010. ISSN 1548-7660. doi: 10.18637/jss.v033.i09. URL `http://www.jstatsoft.org/v33/i09`.

Terry M Therneau. *A Package for Survival Analysis in S*, 2015. URL https://cran.r-project.org/package=survival.

Stephane Vannitsem, Daniel S. Wilks, and Jakob Messner. *Statistical postprocessing of ensemble forecasts*. Elsevier, 2018. ISBN 9780128122488.

W N Venables and B D Ripley. *Modern Applied Statistics with S Fourth edition*. 2002. ISBN 0387954570. doi: 10.2307/2685660.

Laura Verkerk. *Forecasting Infectious Disease Epidemics*. Master's thesis, Leiden University, 2018. URL https://www.universiteitleiden.nl/binaries/content/assets/science/mi/scripties/statscience/2017-2018/2018_06_29_masterthesis_verkerk.pdf.

Mohammad Wasef Hattab. A derivation of prediction intervals for gamma regression. *Journal of Statistical Computation and Simulation*, 2016. ISSN 15635163. doi: 10.1080/00949655.2016.1169421.

Hadley Wickham. The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software*, 40(1):1–29, 2011. URL http://www.jstatsoft.org/v40/i01/.

Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2017. URL https://cran.r-project.org/package=tidyverse.

Rainer Winkelmann. *Econometric analysis of count data*. 2008. ISBN 9783540776482. doi: 10.1007/978-3-540-78389-3.

Alain F Zuur, Joseph M Hilbe, and Elena N Ieno. *A beginner'g guide to GLM and GLMM with R: a frequentist and Baysian perspective for ecologists*. 2013. ISBN 978-0-9571741-3-9.