BEEP V1.0

Bayesian Estimation of Epidemiological Parameters

C. M. Pooley†[1], I. Hinder[2], R. Bailey[3], R. Williams[4], S. Catterall[4], A. Doeschl-Wilson[3] and Glenn Marion[1]

[1] Biomathematics and Statistics Scotland, James Clerk Maxwell Building, The King's Buildings, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, UK.

[2] The University of Manchester, Oxford Rd, Manchester, M13 9PL, UK.

[3] The Roslin Institute, The University of Edinburgh, Midlothian, EH25 9RG, UK.

[4] University of Bristol, Queen's Building, University Walk, Clifton BS8 1TR, UK.

† Corresponding author

THIS DOCUMENT IS CURRENTLY UNDER DEVELOPMENT. WE APPOLOGISE FOR ANY INACCURACIES OR INCOMPLETENESS.

Table of Contents

1 Introduction

This manual is structured in the following way: the introductory section describes what compartmental models are and sketches out the main features of BEEP, along with an easy-to-follow guide for performing a simple analysis. Section 2 sets out, from a mathematical perspective, the models and types of data that can be incorporated into an analysis, and provides some details on how inference is performed. Section 3 focuses on the practicalities of how models and data are specified in an initialisation file used to set up an analysis in BEEP. Section 4 looks at the various options available when running BEEP, and, finally, section 5 describes the outputs that BEEP generates. Some additional example applications are provided in section 6 to illustrate the versatility of this software.

1.1 Compartmental epidemiological models

Compartmental models have widely been used to help understand the collective dynamics of interacting agents, with notable applications in epidemiology and ecology (*e.g.* see [1]).

Figure 1. A compartmental model. This describes how an individual's infection status changes over time. S: susceptible, E: exposed, I: infectious, R: recovered. Individuals are infected at a rate given by the force of infection λ, undergo a non-infectious incubation period given by a gamma distribution Γ (with mean time m and shape parameter k), and finally recover with mean time r.

A simple example is shown in Fig. 1. Here individuals start in the susceptible S compartment and become infected with probability per unit time given by the force of infection λ (which itself will depend on the number of infected individuals in the population plus any infections coming from outside). An infected individual makes the transition to the exposed E state, which means they are infected but not yet infectious. Based on immune system processes, the individual will remain in E (for a gamma distributed period) until they become infectious I, and then, finally, recover R.

Numerous levels of complexity can be added to this basic model, such as: 1) inclusion of various degrees of infection severity (*e.g.* asymptomatic states, all the way up to hospitalised and dead), 2) demographic structure (*e.g.* to capture differences in susceptibility or allow for different age groups in the population to interact with each other to a greater or lesser extent), 3) spatial variation (*e.g.* to account for geographical variation in disease prevalence and transmission rate), 4) temporal variation in disease transmission (*e.g.* to account for social distancing), 5) different strains of pathogen, and 6) the effect of vaccination. Incorporating these various possibilities is discussed in sections 2 and 3.

Many disease control strategies are guided by compartmental epidemiological models. More specifically, these models can be useful to simulate infection dynamics in populations, infer epidemiological parameters from disease data, and predict future outcomes.

1.2 This software

BEEP (Bayesian Estimation of Epidemiological Parameters) is a general-purpose software tool for performing population-based analysis on compartmental epidemiological models.

To clarify terminology, a "compartment" represents the discrete status of an individual at any given point in time (*e.g.* S, E, I and R are all compartments in Fig. 1), a "transition" represents the movement of an individual between two compartments (denoted by the arrows in Fig. 1), "model parameters" are all those quantities in the model that determine how individuals behave (*e.g.* in Fig.1 this would include m, k and r, as well as other parameters incorporated into λ) and the system "state" represents the conglomeration of transition times for all individuals over a time period under consideration.

BEEP incorporates three modes of operation:

Simulation – Given a set of model parameters, potential realisations of the state can be sampled from the model (note, compartmental models are inherently stochastic, so random differences in disease transmission naturally lead to differences in epidemic outcome).

Inference – This is the method by which model parameters are estimated from available data (along with associated uncertainties in these estimates). BEEP accepts a variety of different date types: time series measurements giving transition numbers between selected compartments (*e.g.* daily cases or weekly deaths), populations in different compartments (*e.g.* hospitalised population measured each week) and marginal data (*e.g.* distribution for total number of deaths for different age groups).

Prediction – Based on the results of inference, predictions from the model can be made. These can either be estimates of future behaviour (scenario analysis), or can be used to look at how things would have turned out differently had the model been altered in some specified way (otherwise known as counterfactual analysis).

Once run, BEEP displays outputs using a easy-to-use graphical interface. These outputs may include time variation in state variables, estimates for posterior probability distributions or diagnostic information.

Epidemiological model features:

- Specify arbitrary compartmental epidemiological models.
- Capture time-variation in reproduction number R_t and external force of infection.
- Incorporate spatial stratification.
- Split population into arbitrary demographic classifications (*e.g.* age and/or sex).
- Incorporate susceptibility variation for different demographic groups.
- Incorporate area-based covariates that modify the force of infection, either fixed effects (*e.g.* population density) or time-varying effects (*e.g.* temperature).
- Incorporate a user specified age-mixing matrix (along with potential time modification).
- Specify a matrix for mixing between different areas (along with potential time modification).
- Perform predictions, counterfactuals as well as posterior predictive checks.

Data features:

- Accepts a variety of different data types (from transitions, populations and marginal distributions).
- Incorporate splines to relate measured data to system properties (*e.g.* to account for the fact that only a fraction of true cases are observed).

Software implementation features:

- Efficient parallel implementation (written in C++ with MPI).
- Choose from 8 different inference algorithms.
- A web browser visualisation tool for viewing results on maps, graphs, histograms and tables.

1.3 Getting started

This software is designed to work in a Linux environment and can be used both on a personal computer as well as on high performance computing (HPC).

STEP 1: Download – The code can be downloaded from GitHub using:

```
git clone --recursive https://github.com/ScottishCovidResponse/BEEP.git
```

STEP 2: Load MPI - You need to have MPI installed to compile and run this code. This can be loaded using the command:

```
module load mpi/openmpi-x86_64
```

STEP 3: Build – The project is built using cmake (minimum version 3.13 required). Navigate to the "BEEP" directory and use the command:

```
cmake -Bbuild
```

STEP 4: Compile – The code is compiled on your platform using:

```
cmake --build build
```

BEEP is now ready to use.

1.4 A five-minute guide

Here we provide a five-minute introduction to demonstrate how the software works in a relatively simple scenario. This consists of the following steps:

1.4.1 Simulation example

- Type the following command into the terminal:

  ```
  build/bin/BEEP inputfile="examples/EX.toml" mode="sim"
  ```

  This runs BEEP using the example initialisation file "EX.toml" (shown later in §3.1). This file sets up an SEIR model (Fig. 1) and specifies the outputs that need to be displayed.
- Once run, the terminal output looks like that shown in Fig. 2.
- The directory "examples/Output_EX" contains the outputs from the analysis.
- Opening the file "visBEEP.html" in a web browser allows the user to visualise the results. Figure 3(a) shows an illustrative screenshot providing a summary of the model. Here the user navigates using the menu on the left to view different outputs (see Fig. 4).
- In the sub-directory "Simulate_data" the data files "E-I.csv" and "R.csv" are created (see Fig. 5). These are simulated time series data for the weekly number of cases (*i.e.* number of individuals which undergo the transitions from exposed E to infectious I) and total number of recovered individuals.

Figure 2. Terminal output for a simulation. This shows the terminal output from BEEP when running the example in §1.4.1. First, the file "population.csv" is loaded (which provides information about the population under study, in this case a single area containing 100,000 individuals). Then the actual simulation is carried out (a summary of compartmental populations at selected time points is shown). Finally, the simulated data files "E-I.csv" and "R.csv" are generated in the output directory (see Fig. 5), as well as "visBEEP.html", which provides a point-and-click visual interface to display the results (see Fig. 3(a)).
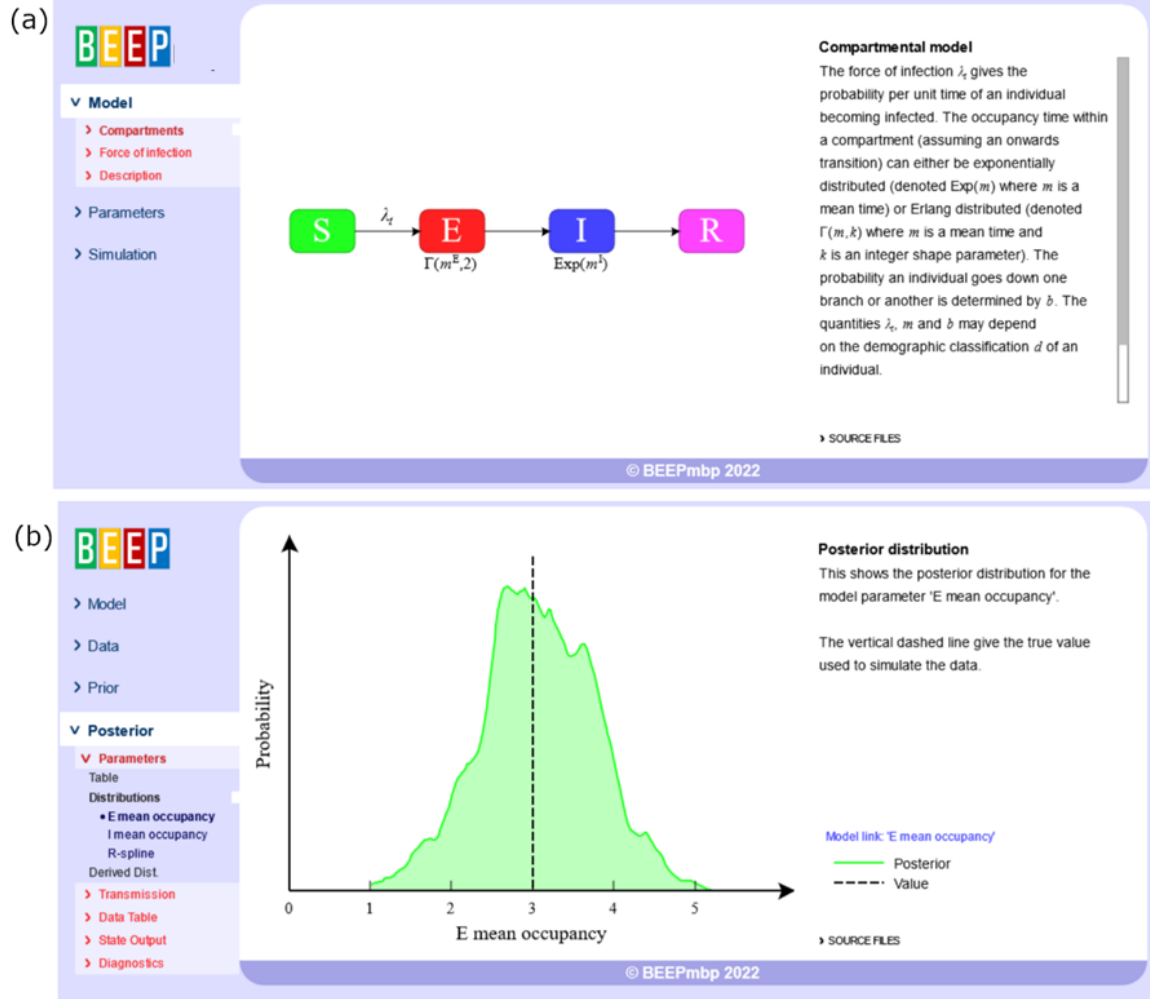
Figure 3. Visualisation. This shows an example of the visualisation software used in BEEP (this is viewed by opening the visBEEP.html file in the output directory). (a) Simulation - The menu on the left allows the user to navigate to view details of the model, parameters and dynamic outputs (see Fig. 4 for examples). (b) Inference - Further tabs can be selected to provide details about the data, prior and posterior (see Fig. 7).

Figure 4. Simulation from model. This shows outputs from BEEP when simulating from the SEIR in Fig. 1. (a) The reproduction number over time (which here has a constant value of 2). (b) The effective reproduction number (which starts at 2, but drops down as the epidemic progresses because of depletion of the susceptible population). (c) The external force of infection (which is taken to have a constant value resulting in 0.5 infected individuals per unit time coming from outside the system under study). (d) The red curve shows the daily number of cases (*i.e.* the rate of E to I transitions) and the black curve shows the derived simulated dataset (which gives the weekly number, leading to this curve's stepped appearance). (e) The red curve shows the number of recovered individuals and the black curve gives this value at weekly intervals (used as data). (f) The dynamic variation in each of the compartmental populations.

Figure 5. Simulated data files. (a) The data file "E-I.csv" generated in §1.4.1. This example gives the weekly number of cases (exposed to infectious transitions) measured from the day in the first column up until just before the day in the row below (hence there were 2 cases between the beginning of day 0 and the end of day 6). (b) The data file "R.csv" gives the number of recovered individuals, recorded at weekly intervals. Note, this example measures time in days but alternatively dates can be used.

1.4.2 Inference example

We now look to perform inference on the simulated data files "E-I.csv" and "R.csv" generated in the previous section (note, case these files have been placed in the "Data_EX" directory, but if BEEP

can't find them there it automatically searches the "Simulated_data" directory in the output directory).

Type the following to run inference (note the "-n 10" refers to the number of CPU cores used and can be changed depending on your computer's architecture):

```
mpirun -n 10 build/bin/BEEP inputfile="examples/EX.toml" mode="map"
```

- Here 'mode' indicates the type of inference algorithm used (see §4.2), MAP is the case.
- The code takes a few seconds to run and the terminal output is shown in Fig. 6.
- Figure 3(b) shows an illustrative screenshot from "visBEEP.html". Note, here we now have menu items on the left to view information about the data, prior and posterior.
- Results from the posterior are shown in Fig. 7. These demonstrates that based on data from weekly cases and the recovered population we are able to make good estimates for the reproduction number and mean time individuals remain in the exposed and infectious compartments (otherwise known as "residency" times).
- Looking at "Parameter_estimates.csv" in Fig. 8 we see that BEEP outputs posterior estimates for all model parameters.

```
Loaded table 'population.csv'.
Loaded table 'E-I.csv'.
Loaded table 'R.csv'.
Number of  model parameters: 5
Number of parameters to be inferred: 3

Running....

By default generations are iterated until convergence.
By default 'nparticle' set to 20.
By default 'posterior_particle' is set to 20.
By default 'nsample_final' is set to 400.

Start Inference...
Generation 0 - Best log(Posterior prob): -134.626
Generation 1 - Best log(Posterior prob): -118.424
Generation 2 - Best log(Posterior prob): -110.915
Generation 3 - Best log(Posterior prob): -115.437
Generation 4 - Best log(Posterior prob): -112.518
Generation 5 - Best log(Posterior prob): -109.646
Generation 6 - Best log(Posterior prob): -109.804
Generation 7 - Best log(Posterior prob): -108.581
Generation 8 - Best log(Posterior prob): -108.471
Generation 9 - Best log(Posterior prob): -108.346
Generation 10 - Best log(Posterior prob): -108.35
Generation 11 - Best log(Posterior prob): -108.222
Generation 12 - Best log(Posterior prob): -108.315
Generation 13 - Best log(Posterior prob): -108.224
Generation 14 - Best log(Posterior prob): -108.225
Generation 15 - Best log(Posterior prob): -108.224
Generation 16 - Best log(Posterior prob): -108.21
Generation 17 - Best log(Posterior prob): -108.209
Generation 18 - Best log(Posterior prob): -108.207
Generation 19 - Best log(Posterior prob): -108.207
Generation 20 - Best log(Posterior prob): -108.207
Generation 21 - Best log(Posterior prob): -108.207
Generating posterior samples...
Scale covariance...
SMC sampler...
Output results...
Gathering samples...

Generating outputs in directory 'examples/Output_EX'...
  Posterior parameter estimates given in 'Parameter_estimates.csv'
  Open 'visBEEP.html' to visualise results.
WARNING: The TOML command 'modification' is not used.

Total time: 0.0520 minutes.
```

Figure 6. Terminal output under inference.

This show the terminal output from BEEP when running the example in §1.4.2. First the file "population.csv" is loaded, which provides information about the population under study. The data files "E-I.csv" and "R.csv" are loaded, providing information about weekly cases and the overall recovered population. Inference is then performed, and what is displayed on the terminal will depend on the inference algorithm selected. In this case MAP proceeds in a series of generations which successively improve the maximum *a posteriori* estimate until no further improvement can be made. The file "Parameter_estimates.csv" provides posterior estimates for model parameter (Fig. 8) and "visBEEP.html" is created to visualise the results (Fig. 3(b)).
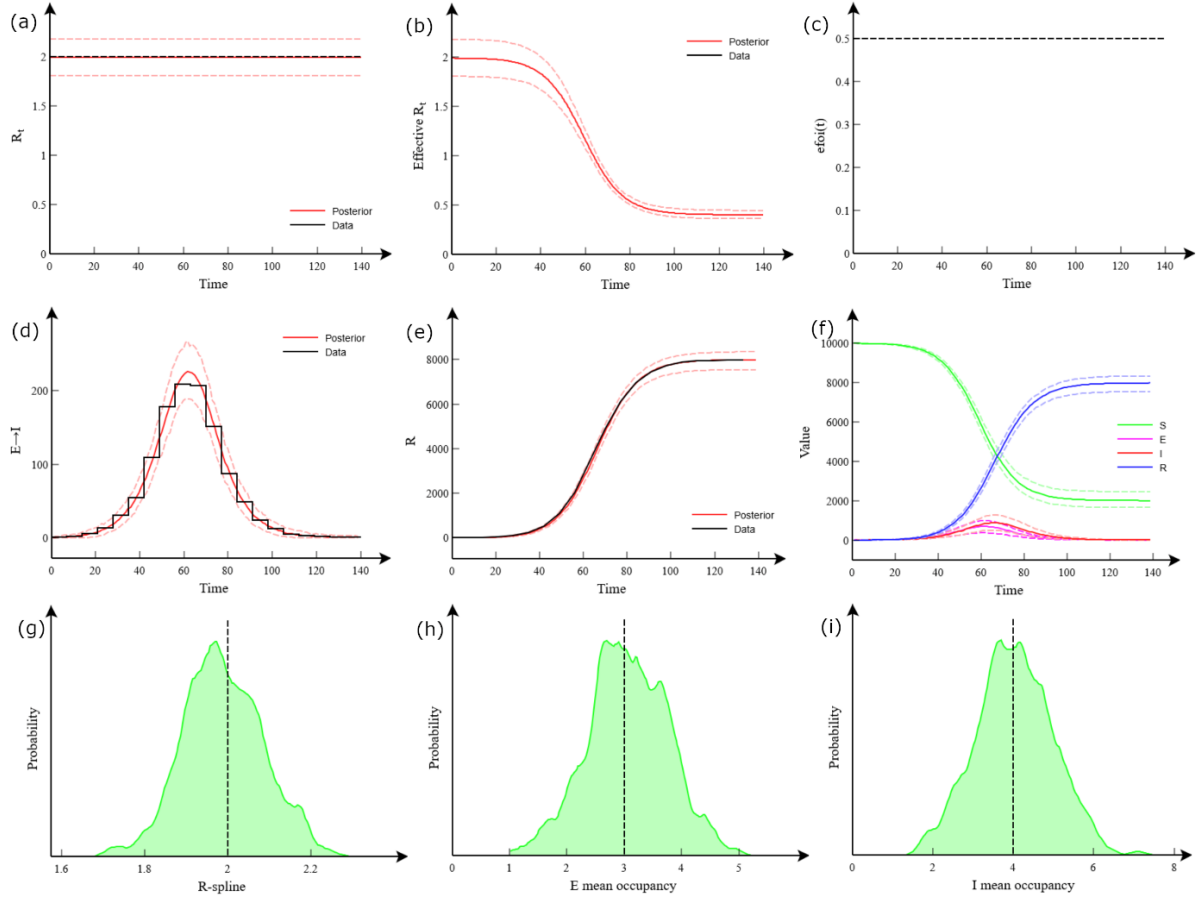
Figure 7. Inference from model. This shows some of the outputs from BEEP when performing inference using the data files in Fig. 5. (a) The posterior distribution for the reproduction number R_t, with the dashed lines representing 95% credible intervals. (b) The effective reproduction number R_t^{eff}. (c) The external force of infection (assumed to be constant). (d) The red curve shows the inferred rate of cases and the black curve shows the fit with the actual data. (e) The red curve shows the inferred population in the recovered state and the black curve shows the fit with the actual data. (f) The posterior distributions for the inferred dynamic variation in each of the compartmental populations. (g-i) Posterior probability distributions for three model parameters: R_t and the mean residency times in the exposed E and infectious I compartments. The black dashed lines show the true simulated values, which are contained within the credible intervals, indicating successful inference.

	A	B	C	D	E
1	Name	Mean	95% CI	SD	Prior
2	E mean occupancy	3.054	1.843 --- 4.179	0.601	Uniform(1|10)
3	I mean occupancy	4.045	2.499 --- 5.555	0.783	Uniform(1|10)
4	I Infectivity	1	1.000 --- 1.000	0	Fixed(1)
5	R-spline	1.99	1.814 --- 2.176	0.092	Uniform(0.5|4)
6	efoi-spline	0.5	0.500 --- 0.500	0	Fixed(0.5)

Figure 8. Parameter estimates. BEEP generates the file "Parameter_estimates.csv" in the output directory. This file provides posterior estimates for model parameters. The different columns give: the parameter name, posterior mean, 95% credible interval, standard deviation and prior (specified in the input TOML file). In this example two parameters are fixed: the infectivity of individuals in the I compartment (arbitrarily set to one), and the external force of infection (set to a small value sufficient to initiate an epidemic).
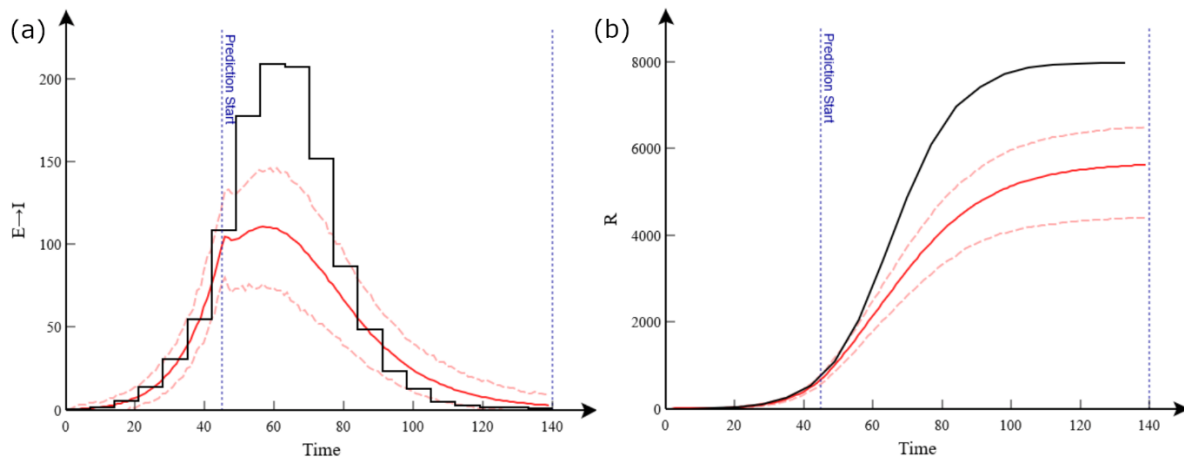
Figure 9. Prediction. These plots predict the effect of modifying the model such that the transmission rate is reduced by 30% after day 45, *e.g.* mimicking the impact of a lockdown. (a) The red curve gives the modified posterior distribution for the case rate. After the prediction start time there is a marked reduction compared to the simulated dataset. (b) The modified posterior distributions for the population in the recovered compartment.

1.4.3 Prediction example

Whenever inference is performed posterior samples are stored in the "Posterior/samples/" directory. These samples can subsequently be used to make model predictions:

- Type the following to run prediction:

  ```
  mpirun -n 10 build/bin/BEEP inputfile="examples/EX.toml" mode="prediction"
  ```
  In this case a modification to the model (as specified in "EX.toml") is assumed that reduces the transmission rate after day 45 (*e.g.* this could be used to represent lockdown in the case of a pandemic).
- Results from the "visBEEP.html" file are shown in Fig. 9. This reveals a substantial reduction in the severity of the pandemic as a result of the intervention.

The example application described above relied on many simplifying assumptions which may not be valid in real world scenarios (*e.g.* infection events are not usually all known and R_t typically exhibits large temporal variation that needs to be accounted for). Nevertheless, this section has served as a useful starting point to introduce the software. Further examples are provided in §6 that explore how these additional complications can be incorporated into the model.

2 Models, data and inference

In this section we provide a broad mathematical description of the sorts of model used in BEEP. In the next section we go on to describe, in detail, how these models are implemented in practice using a TOML initialisation file.

2.1 Epidemiological model

2.1.1 Stratification of the population

BEEP assumes the population under study can potentially be stratified in three different ways:

Disease status stratification – The variable c is used to denote different compartments that specify an individual's disease status (*e.g.* in Fig. 1 this can either be S, E, I or R). BEEP allows the user to specify an arbitrarily number of compartments in the model. For example, as well as susceptible, exposed, infectious and recovered compartments, others can be added to represent asymptomatic, pre-symptomatic or hospitalised individuals.

The residency time within each compartment (that is the time between when an individual enters and leaves a compartment) is restricted to either be exponentially distributed[1] or Erlang distributed[2], with a model parameter setting its mean. In cases in which the compartmental model branches, further parameters are associated with the probability for individuals to pass down one branch or another.

Demographic stratification – The variable d is used to denote the demographic grouping an individual belongs to. In the simplest case this would consist of a single category such that, from an epidemiological point of view, everyone in the population behaves the same. On the other hand, the population can be split by age, sex, or any other discrete classifier. Parameters determining the mean residency time and branching probabilities from above can be made dependent on these demographic groupings (to reflect the fact that the disease affects different groups in different ways).

Geographic stratification – The variable a is used to indicate the area in which an individual resides. For a non-spatial model the system would consist of just a single area (*e.g.* a nationwide analysis), but if we are interested in accounting for the geographic spread of disease then a could represent different areas, *e.g.* local authorities.

As well as stratification of the population, it is also possible to divide the infectious agents into different "strains", indexed by s.

2.1.2 The force of infection

Whilst arbitrary compartmental model specification is allowed in BEEP, only a single specified transition is associated with infection (in Fig. 1 this is the S→E transition). In the general case, the

---

[1] An exponentially distributed residency time is equivalent to there being a certain constant probability per unit time of leaving a compartment, consistent with a Poisson process.
[2] The Erlang distribution is the same as the Gamma distribution, but restricted to have integer shape parameter. If k=1 it is the same as an exponential distribution, but if k is large the residency time becomes more peaked around its mean with an almost normal distribution.

probability per unit time of an individual becoming infected with strain s, in area a, in demographic group d at time t is given by the force of infection:

$$\lambda_{s,a,d,t} = \sigma_d \left[r_{s,t} R_t \psi_s \alpha_{a,t} \left[\sum_{a',d',c} M_{a,a',t} A_{d,d',t} i^c N^c_{a',d',t} \right] + \tfrac{1}{N} \eta_{s,a,d,t} \right]. \tag{1}$$

The various terms in this expression are explained below:

Susceptibility – σ_d gives the relative susceptibility of demographic group d. This itself is decomposed into separate susceptibilities for each demographic classification q in the model (*e.g.* q could be "Age" or "Sex") multiplied together, *i.e.*

$$\sigma_d = \prod_q \sigma_{q,d_q}, \tag{2}$$

where $v=d_q$ gives the value of classification q for demographic group d (*e.g.* if a particular d represents males in the age range 40-59 then the values for its classifications are $d_{Age}=$"40-49" and $d_{Sex}=$"Male") and $\hat{\sigma}_{q,v}$ is the relative susceptibility for that value. Because $\hat{\sigma}_{q,v}$ represents a *relative* susceptibility, its values are constrained by

$$\sum_v \sigma_{q,v} \phi_{q,v} = 1 \quad \text{for each } q, \tag{3}$$

where $\phi_{q,v}$ is the overall proportion of individuals with value v in classification q. This ensures that for each classification the average susceptibility over the entire population is exactly one. See §3.5.3 for implementation of susceptibility variation in BEEP.

Reproduction number – The quantity R_t represents the reproduction number as a function of time, which is defined to be the expected number of secondary cases directly generated by a single case in a population in which nearly all individuals are susceptible (see §3.4.3 for implementation). Estimation of this quantity relies on the approach taken by Diekmann *et* al. [2], in which R_t is calculated from the highest eigenvalue of the next generation matrix (see Appendix A for more details).

Incorporation of R_t into Eq.(1) makes sense because it would be expected that disease transmission within the population should to be proportional to the rate at which effective disease transmitting contacts occur ($r_{s,t}$ is a quantity derived from other model parameters which sets this proportionality constant[3], again see Appendix A). In reality, due to depletion of the susceptible population, the actual number of secondary cases directly generated by a single case is given by the effective reproduction number R^{eff}_t. This quantity that can be derived from R_t and other model parameters (Appendix A) and is displayed as an output by BEEP.

Strain effect – If more than one disease strain exists in the system, the factor ψ_s in Eq.(1) multiplies the reproduction number R_t to give its value for that particular strain s[4] (see §3.8 for implementation).

---

[3] $r_{s,t}$ essentially gives the time-varying ratio $\beta_{s,t}/(R_t \psi_s)$, where $\beta_{s,t}$ is the transmission rate.
[4] When there is one strain, $\psi=1$, and for more than one strain, $\psi_s=1$ for the reference strain.

Area effects – The term $\alpha_{a,t}$ in Eq.(1) is used to incorporate geographical variation in disease transmission (see §3.6.4 for implementation). This, itself, is decomposed into various potential contributions:

$$\alpha_{a,t} = w_a \times e^{\sum_i X_{a,j} b_j} \times e^{\sum_i X'_{a,t,j} b'_j} \times h_{l_{a,t}}.$$ (4)

The first is a relative area effect w_a, which accounts for the fact that the overall disease transmission in one area might be different to another. This is normalised by

$$\sum_a w_a \phi_a = 1,$$ (5)

where ϕ_a is the fraction of the population in area a, such that the average area effect is one.

The second term in Eq.(4) incorporates area fixed effects. Here X is a design matrix and b is a vector of fixed effects (note, the exponential link function ensures the rate is strictly positive). These are aimed at correlating local transmission rate with measurable factors within each area. For example, suppose column 1 in X gives the log of the local population density p_a. This would mean that Eq.(4) contains a factor given by

$$e^{\log(p_a) b_1} = p_a^{b_1},$$ (6)

where b_1 is a fixed effect that is set to a specified value or inferred (here b_1=0 and b_1=1 correspond to frequency and density dependent transmission, respectively).

The third term in Eq.(4) allows for time-varying fixed effects. The last term aims at estimating the effect of different levels of disease intervention l (for example one level might represent a complete lock down, whereas another might be just social distancing). The matrix $l_{a,t}$ is an input which specifies the level each area a has at any point in time t. h_l represent the relative transmission rate for level l, normalised by

$$\sum_l h_l \phi_l = 1,$$ (7)

where ϕ_l is the fraction of time areas spend in level l (averaged over all areas and times).

Geographical mixing of individuals – The term $M_{a,a',t}$ in Eq.(1) gives the effective contact rate of individuals between areas a and a' at time t (see §3.6.3 for implementation). By default this matrix is assumed fixed, but time variation can be incorporated in the following way:

$$M_{a,a',t} = Z_{a,a'} m_t + D_{a,a'}(1 - m_t),$$ (8)

where $Z_{a,a'}$ is a constant matrix (which is an input into BEEP and, for example, could be based on census commuter data[5], see Appendix C) and $D_{a,a'}$ is a diagonal matrix (with elements given by the reciprocal of the population sizes in the corresponding areas). In Eq.(8) m_t is a time-varying parameter that determines the rate of mixing between areas (m_t=0 corresponds to no mixing and

---

[5] It is important to note that we only need to know the input $Z_{a,a'}$ up to a constant factor.

m_t=1 corresponds to mixing according to $Z_{a,a'}$). If movement restriction are in place we might expect m_t to lie somewhere in between these two extremes.

Demographic mixing of individuals – This is characterised by the matrix $A_{d,d'}$ in Eq.(1) (see §3.5.1 for implementation). Currently BEEP supports only age-dependent mixing factors (since age is the biggest determining factor in characterising mixing patterns in humans). In this case

$$A_{d,d',t} = B_{z,z',t}, \tag{9}$$

where z and z' are the age groups of the demographic groups d and d', respectively, and $B_{z,z',t}$ is a time dependent square matrix with size given by the number of age categories. In the first instance $B_{z,z',t}$ is taken to be a constant matrix $C_{z,z'}$ (which is itself derived from a user defined age mixing matrix, such as that published by POLYMOD [3] or BBC Pandemic [4], see Appendix D for details) . However, time dependency can also be accounted for by defining modifications to this matrix. For example, a factor f_t (defined by a spline) could be incorporated which multiplies one of the rows and columns in $C_{z,z'}$. This allows for the mixing exhibited by one particular age group with the rest of the population to be modulated in time.

Infectivity – i^c gives the relative compartmental infectivity (see §3.4.1 for implementation). This can be used to account for the fact that some compartmental states are more infectious than others (*e.g.* asymptomatic would be expected to be less infectious than symptomatic).

Subpopulation size – $N_{a',d',t}^c$ gives the total number of individuals in compartment c, area a' and demographic group d' at time t^6.

External infections – Finally, $\eta_{s,a,d,t}$ in Eq.(1) gives the spatial and temporal variation in the external force of infection caused by infections being introduced from outside the population under study (see §3.4.4 for implementation).

2.1.3 System dynamics
BEEP uses a τ-leaping algorithm (with fixed time step τ) to approximate a continuous time Gillespie algorithm[7]. We denote $p_{a,d,c,t}$ to be the population in area a, demographic group d, compartment c at time t (note, for the purposes of analysis, strain s is treated as another demographic classification in d).

Transitions within the model (*e.g.* represented by the arrows in Fig. 1) are indexed by j. They move individuals from an initial compartment i_j to a final compartment f_j. Within a time period between t and $t + \tau$ the number of transitions of type j in area a and demographic group d is taken to be Poisson distributed. For the infection transition j_{inf} this is given by

$$n_{a,d,j_{\mathrm{inf}},t} \sim Poisson\left(\tau p_{a,d_{-s},S,t}\lambda_{s,a,d,t}\right), \tag{10}$$

---

[6] Note, the demographic group d specifies the strain these individuals are infected with.
[7] τ can be specified, but by default it is set to 0.25 days.

where S is the susceptible compartment[8], and the notation d_s is used to represent the fact that strain is disregarded as a demographic classifier for S (an individual's strain becomes determined only after they become infected[9]).

For non-infection transitions j, the number is given by

$$n_{a,d,j,t} \sim Poisson\left(\tau\, p_{a,d,i_j,t}\, b_{d,j}\, /\, \mu_{d,i_j}\right), \tag{11}$$

where $b_{d,j}$ is the branching probability for individuals to move down transition j (as opposed to any other transition leaving initial compartment i_j) and μ_{d,i_j} is the residency time in i_j.

As a result of the transitions in Eq.(11), populations needs to be updated:

$$p_{a,d,c,t+\tau} = p_{a,d,c,t} + \left[\sum_{j \in \text{enter } c} n_{a,d,j,t}\right] - \left[\sum_{j \in \text{leave } c} n_{a,d,j,t}\right], \tag{12}$$

where the first sum goes over all transitions that enter compartment c (increasing its population), and the second goes all those that leave c (decreasing its population).

Erlang distributions with shape parameter k and mean residency time m are constructed by sequentially linking together k compartments, each with exponentially distributed residency time that has a mean of m/k.

The initial population sizes $p_{a,d_{-s},c,t_{\text{start}}}$ set the initial conditions. By default all individuals are assumed to start in the susceptible compartment[10] S. Population stratification across different areas a and demographic groups d_s are inputs which must be specified. For the most part these are assumed constant (so, for example, a person's place of residency remains unchanged). However, it is also possible in BEEP to externally impose changes in the demographic stratification of the population. This is especially useful when accounting for vaccination. Here changes are imposed on this system (as inputs) rather than occurring as the result of an underlying stochastic process.

BEEP can also be run in deterministic mode. This is an approximation to the system dynamics which becomes increasingly valid for larger epidemic sizes (and so, for example, it may be sufficiently good to run a nationwide level analysis), and allows for faster inferences to be achieved. In this case, the number of transitions is simply set to the mean of the Poisson distribution rather than being sample from it:

$$n_{a,d,j_{\text{inf}},t} = \tau\, p_{a,d_{-s},S,t}\, \lambda_{s,a,d,t},$$
$$n_{a,d,j,t} = \tau\, p_{a,d,i_j,t}\, b_{d,j}\, /\, \mu_{d,i_j}. \tag{13}$$

2.2 Data
BEEP can accept data of the following types:

---

[8] Here S is defined to be the state from which infection events occur, *e.g.* in Fig. 1 this corresponds to the "S" compartment.

[9] For simplicity, the possibility of an individual becoming infected with multiple strains is ignored.

[10] BEEP does allow for modification to this.

- **Time series transition data** – As in Fig. 4, this is represented by a list giving the number of individuals passing down a specified transition (or multiple transitions) over a series of time steps, *e.g.* every 7 days. Transitions can further be stratified by area or demographic group leading to a table of inputs (*e.g.* different columns could represent geographical areas and different rows represent different weeks). This can be used to incorporate, *e.g.*, case data, hospital admissions or death data.
- **Time series population data** – This is represented by the population in a compartment (or compartments) at a series of time points[11]. Again, this data can be stratified by age or demographic group, and can be used for, *e.g.*, hospital population data.
- **Time series population fraction data** – The same as population data except expressed as a fraction of the entire population. This can be used for, *e.g.*, seroprevelence data[12].
- **Marginal data** – This is represented by the number of individuals going down a specified transition (or multiple transitions) over a specified time period (*e.g.* from the beginning to the end of an analysis period) stratified by area or demographic group. Such data is often publically available, *e.g.* the number of hospital admissions broken down by age group.

2.3 Bayesian inference

Collectively the model parameters are referred to as θ. The initial populations $p_{a,d-s,c,t_{\text{start}}}$ and transitions between states $n_{a,d,j,t}$ define the system dynamics. These are collectively referred to as ξ. In reality, ξ is not known, and from a Bayesian point of view is considered as a set of latent model variables.

Application of Bayes' theorem implies that the posterior probability distribution is given by

$$\pi(\theta,\xi\,|\,y) \propto \pi\left(y\,|\,\xi\right)L(\xi\,|\,\theta)\pi(\theta), \tag{14}$$

where $\pi(y|\xi)$ is the "observation model", which gives the probability of the data given a system state, $L(\xi|\theta)$ is the "latent process likelihood", which gives the probability of the state given a set of model parameters, and, finally, $\pi(\theta)$ is the prior, which captures the state of knowledge regarding parameter values before data y is considered.

BEEP incorporates 8 different inference algorithms which aim to generate samples from the posterior distribution in Eq.(14). These methods can be split into three contrasting approaches:

2.3.1 Maximum *a posteriori* (MAP) estimation

The aim of this approach is to calculate the set of parameter values θ that maximise the *a posteriori* probability $L(y|\theta)\pi(\theta)$. To do this is it necessary to implement a method that estimates the likelihood $\pi(y|\theta)$ directly. In general this is not possible to do exactly, but it can be done approximately. Specifically, it is necessary to assume that the populations in the different compartments are multivariate normally distributed (this approximation is valid provided the number of individual in the system is reasonably large). These distributions then evolve in time according to a deterministic set of equations. Observations on the system also act to modify these distributions, again in a

---

[11] In this case they do not need to be equally spaced in time.
[12] Seroprevalence studies gather random samples of individuals from the population to estimate the current infection rate.

deterministic manner. Integration of the equations over the time period under observation yields an estimate for $L(y|\theta)$ (see Appendix F for details).

Covariance matrix adaptation evolution strategy (CMA-ES) – This algorithm has become a widely-used methodology for numerical optimization of functions, and is found to work well even when the posterior surface is rough (*e.g.* containing many meta-stable state). It proceed over a series of generations which converge on the MAP. CMA-ES also provides an estimate for the Hessian matrix, so this method allows for estimation for parameter uncertainties.

2.3.2 Power posterior

Whilst Eq.(14) represent an ideal, often it is found that generating true posterior samples takes an infeasibly long CPU time, especially when y contains a lot of data[13]. One way around this to consider a modified distribution called the power posterior:

$$\pi_{PP}(\theta,\xi \mid y,\phi_{post}) \propto \left[\pi\left(y \mid \xi \right) \right]^{\phi_{post}} L(\xi \mid \theta)\pi(\theta), \tag{15}$$

where ϕ_{post} is a parameter called the inverse temperature. When $\phi_{post}=1$ the true posterior is recovered and when $\phi_{post}=0$ the system maps of the prior (with disregard for the data). Setting ϕ_{post} less than one can substantial speed up inference, but only marginally affect posterior estimates (essentially justifying this approach).

Four different inference algorithms aim to draw samples from the power posterior in Eq.(15):

Markov chain Monte Carlo with model-based proposals (MCMC-MBP) – This approach runs an MCMC chain at a specified ϕ_{post} using so-called model-based proposals (MBPs). MBPs allow for joint proposals in θ and ξ to explore the power posterior more efficiently than changes made in conventional data augmentation schemes[14].

Metropolis coupled MCMC (MC³) – Here K separate MCMC chains, indexed by k, are run in parallel (again, making use of MBPs). The inverse temperatures of these chains are selected to span from the posterior (k=1) to the prior (k=K) [15]:

$$\phi_k = \left(\frac{K-k}{K-1} \right)^4 \phi_{post}. \tag{16}$$

This approach has the advantage of improved mixing[16] (states can be swapped between chains), is less prone to getting stuck in metastable states (the higher temperature chains effective smooth out the posterior landscape allowing for a greater degree of exploration) and can be used to estimate the model evidence (see §2.3.6 later). However, this method does come with the addition computational burden of running multiple chains.

---

[13] When the data is very restrictive on the states the system can accept, this can lead to long mixing times and/or the system becoming stuck in metastable states.

[14] These would typically make small changes to each of the quantities in $n_{a,d,j,t}$ and accept or reject these changes based on a Metropolis-Hastings probability.

[15] Note, the power 4 is found to empirically work well under most scenarios. This factor, however, can be specified in the software using the 'invT_power' command.

[16] Good mixing implies that consecutive samples along the MCMC chain are less correlated.

Particle annealed importance sampling using MBPs (PAIS-MBP) - PAIS-MBP starts with a number of randomly sampled "particles" from the prior. The algorithm proceeds through a series of generations and within each generation: a) the inverse temperature is increased such that, on average, half of the particles are copied and half are culled, and b) a series of MBPs allow particles to explore parameter and state space (to avoid degeneracy).

Particle-MCMC (PMCMC) – This runs a single MCMC chain at a specified inverse temperature ϕ_{post}, but here the state ξ is constructed separately for each Metropolis-Hastings proposal. This construction requires a sequential Monte Carlo filtering step that uses multiple parallel simulations from the model, otherwise known as particles, along with filtering of these particles at predetermined time intervals. When large amounts of data are analysed, this method requires a large number of particles to efficiently run, which can lead to very computationally slow proposals. However, PMCMC tends to mix faster than other approaches, helping to mitigate this effect.

2.3.3 Approximate Bayesian computation

Rather than use the full observation model in Eq.(14), approximate Bayesian computation (ABC) relies on introducing a measure the fit between the data y and the state ξ called the error function (EF). We define a model "sample" to be generated by first sampling the model parameters θ from the prior $\pi(\theta)$ and then simulating the state ξ from the model. Under ABC the posterior distribution is approximated by sampled states θ and ξ for which EF is below a given cut-off value EF_{cutoff}. This can be represented by

$$\pi_{ABC}(\theta, \xi \mid y, EF_{cutoff}) \propto H\left(EF(y \mid \xi) - EF_{cutoff}\right) L(\xi \mid \theta)\pi(\theta), \qquad (17)$$

where H is the Heaviside step function.

For convenience we assume that the error function is functionally related to the observation model through[17]

$$EF(y \mid \xi) = -2\log\left(\pi\left(y \mid \xi\right)\right). \qquad (18)$$

BEEP incorporates three algorithms for generating samples from Eq.(17):

Approximate Bayesian Computation (ABC) – This generates posterior samples by means of a simple ABC rejection sampling scheme. This sample from the model (as described above) and accepts only those samples with error function below the cut-off.

Approximate Bayesian Computation using sequential Monte Carlo (ABC-SMC) – This runs over a number of generations, with the previous generation being used as an importance sampler for the next. Under some circumstances this approach is significantly more computationally efficient than the standard ABC algorithm.

Approximate Bayesian Computation with model-based proposals (ABC-MBP) – This starts with a number of randomly sampled "particles" from the prior (a particle comprises of a parameter set and a system state). The algorithm proceeds through a series of generations and within each generation:

---

[17] This definition implies that a normally distributed observation model transforms to a sum of residuals error function.

a) the EF cut-off is reduced such that half the particles are copied and half are culled, and b) a series of MBPs allow particles to explore parameter and state space (to avoid degeneracy).

2.3.4 Observation models

Here we list different possible observation models that BEEP can use to relate the observed data to the equivalent quantity derived from the state ξ (note through the relationship in Eq.(18) this also defines potential error functions that can be used).

For simplicity BEEP assumes that observations are independent[18], and so

$$\pi(y \mid \xi) = \prod_i \pi(y_i \mid Y_i(\xi)), \tag{19}$$

where i indexes measurements, y_i is the value of the ith measurement (*e.g.* this could come from one of the table entries in Fig. 4) and Y_i is the equivalent value derived from the state ξ. The possible choices are:

- **Normal** – The data is normally distributed around the state value with a specified standard deviation:

$$\pi(y_i \mid Y_i(\xi)) = N(y_i \mid Y_i(\xi), \sigma_i), \tag{20}$$

  The value of the standard deviation σ_i can either be set separately for each observation or be constant across a given data source. Furthermore, it can either defined in absolute terms or as a percentage of the observation. In the latter case the following relationship is used:

$$\sigma_i = 0.01 \times p_i \left(y_i(1-\varepsilon) + y_{\max}\varepsilon \right), \tag{21}$$

  where p_i is the percentage, y_i is the observed value, y_{max} is the maximum value for a given time series, and ε is a small constant (set to 0.05 by default) which ensures that σ_i does not become zero when y_i becomes zero.

- **Poisson** – The data is Poisson distributed around the state value:

$$\pi(y_i \mid Y_i(\xi)) = P(y_i \mid Y_i(\xi)), \tag{22}$$

  where $P(x|\mu)$ is the Poisson probability distribution for x given a mean μ.
- **Negative binomial** – The data has a negative binomial distribution around the state value:

$$\pi(y_i \mid Y_i(\xi)) = NB(y_i \mid Y_i(\xi), k), \tag{23}$$

  where $NB(x|\mu,r)$ is the negative binomial probability distribution for x given a mean μ and shape parameter r. This parameterisation implies a variance of $\mu + \mu^2/r$, and so the negative binomial is over-disperse when compared to a Poisson distribution (which has a variance of just μ).

---

[18] This assumption is valid for transition data, but less so for population data (*e.g.* the hospital population on one day will clearly be strongly correlated to that from the previous day).

The options above can be selected separately for each of the data files loaded into BEEP. Incorporation of thresholds into the model is discussed in Appendix E.

2.3.5 Incomplete observations

For some measureable quantities we can be confident that the values from data are directly comparable into the data types in §2.2, *e.g.* statistics on hospitalisations and deaths. However for other measureable quantities things become less clear. Consider the daily number of disease cases. Here it is tempting to directly relate these to the daily number of infection transitions that occur in the model. This, however, might represent a huge underestimate, because observed cases heavily depend on rate at which individuals are tested (also tests are imperfect and do not pick up all infections)[19] . This is further complicated by the fact that testing effort may have changed over time.

To help mitigate this, BEEP allows for $Y_i(\xi)$ to be modified in such a way that it can be used to directly compare to the data y_i. In particular a time-dependent scaling can be added:

$$Y_i' = s_t Y_i, \tag{24}$$

where s_t is a time-varying spline (which can either be specified or inferred). This modified value is then used in the observation model in §2.3.4.

Considering the daily case data discussed above, this would first need be shifted in time[20] (to account for the difference in time between becoming infected and becoming a case), and then it could be directly compared to the modified $Y_i(\xi)$ (using Eq.(24) which accounts for time variation in testing availability) through the observation model.

2.3.6 Model evidence

The model evidence (ME) $\pi(y)$ is a measure for comparing how well the data y agrees with the model[21]. One way to define this is as the expected value of the observation model over random samples from the model ξ [22]:

$$\pi(y) = \left\langle \pi(y \mid \xi) \right\rangle_\xi \tag{25}$$

Models that generate samples in good agreement with the data have a higher ME (because their observation model probability is higher) and *vice versa*.

As discussed above, for most practical applications it is not possible to sample from the true posterior. Following sections 2.3.2, the ME for the power posterior is given by

$$\pi(y \mid \phi_{post}) = \left\langle \left[\pi(y \mid \xi) \right]^{\phi_{post}} \right\rangle_\xi, \tag{26}$$

---

[19] The test sensitivity *Se* characterises the probability that a truly infected individual generates a positive test result.

[20] When time series data is loaded into BEEP it can be shifted in time by setting the 'shift' property.

[21] Other model selection techniques such as DIC have proven to be notoriously unreliable for the sorts of epidemiological model considered here.

[22] These are generated by first sampling model parameters θ from the prior $\pi(\theta)$ and then simulating the state ξ from the model.

and for ABC approaches in §2.3.3

$$\pi(y \mid EF_{\text{cutoff}}) = \left\langle H\left(EF(y \mid \xi) - EF_{\text{cutoff}}\right)\right\rangle_{\xi}. \tag{27}$$

It is important to note that the ME is not just a single value, but depends on either the inverse temperature of the prior ϕ_{post} or error function cut-off EF_{cutoff}. When performing model comparison, therefore, care must be taken to ensure it is done so at a consistent value of ϕ_{post} (or EF_{cutoff}). The ratio in ME gives the Bayes factors between models. A Bayes factor above 3.2 is considered substantial support for one model over another, and over 10 provides strong evidence (Kass and Raftery (1995)).

Appendix G describes how the ME is calculated for the algorithms which can estimate this quantity[23].

---

[23] PMCMC and MCMC-MBP are not able to estimate model evidence because they only explore parameter space around the posterior.

3 The input TOML file

The input TOML file contains the information BEEP requires in order to define the model (from §2), reference data files, and perform simulation, inference or prediction. An index of all TOML commands is given in a glossary at the end of the manual. BEEP also contains numerous examples, as discussed in §6, which help to illustrate practical uses for this software.

3.1 A simple example

We first go through the following very simple example (taken from the file "examples/EX.toml" and first introduced in §1.4):

```
time_format = "number"
start = "0"
end = "140"
datadir = "examples/Data_EX"
outputdir = "Output_EX"
comps = [{name="S"},
          {name="E", dist="Erlang", k="2", mean_value="4"},
          {name="I", dist="Exp", mean_value="4", inf_value="1"},
          {name="R"}]
trans = [{from="S", to="E", infection="yes"},
         {from="E", to="I"},
         {from="I", to="R"}]
R_spline = [{value="2.0", prior="Uniform(0.4,4)"}]
efoi_spline = [{value="0.5"}]
areas = "population.csv"
data_tables = [
   {type="transition", observation="E->I", timestep="7", obsmodel="normal 10%", file="E-I.csv"},
   {type="population", observation="R", timestep="7", obsmodel="normal 10%", file="R.csv"}
]
state_outputs = [
      {plot_name="Dynamics", type="population", observation="S", line_colour="green"},
      {plot_name="Dynamics", type="population", observation="E", line_colour="yellow"},
      {plot_name="Dynamics", type="population", observation="I", line_colour="red"},
      {plot_name="Dynamics", type="population", observation="R", line_colour="blue"}
]
modification = [{start="30", type="beta_fac", factor="0.3"}]
```

'time_format' determines how time is represented (in both the TOML file and also any data files). This can take the values "*year-month-day*", "day/month/year", "day.month.year" or "number", and is set depending on the type of data available.

'start' and 'end' give the start and end dates/times for analysis.

'datadir' defines the location of the data directory.

'outputdir' indicates where outputs from BEEP are placed.

'comps' defines the compartments in the model, which in this case are specified to be: susceptible S, exposed E, infectious I and recovered R (see Fig. 1). 'dist' specifies the distribution given to the residency time, where "Exp" denotes the exponential distribution and "Erlang" denotes the Erlang distribution (with corresponding shape parameter k). Under simulation the means of these distributions are specified by 'mean_value'. If no distribution is set, it is assumed that no transitions leave the compartment[24]. The relative infectiousness is set by 'inf_value', and if omitted is assumed zero[25].

'trans' defines the transitions in the model (corresponding to the arrows in Fig 1). Setting 'infection' to "yes" indicates the infection transition (only one such transition can be defined in this way). The rate of this particular transition is determined by the force of infection in Eq.(1).

'R_spline' defines a spline that describes how the reproduction number R_t in Eq.(1) changes over time (see §3.3 for how splines are defined). In this particularly simple case, its value is set to be constant. The prior is set to a uniform distribution between 0.4 and 4, which means under inference the spline that specifies R_t is bounded between these two values.

'efoi_spline' defines a spline giving the external force of infection $\eta_{s,a,d,t}$ Eq.(1). This is used to generate infections in the system that initiate epidemics in the first place.

'areas' specifies a file giving information about the population under study. This file must reside in the 'datadir' directory and can either be in ".csv" format (in which case columns are separated by commas) or ".txt" format (columns are tab separated). Here the file "population.csv" is:

```
area,population
Region,10000
```

One column in this file must have the heading "area", and this to refers the geographical area under consideration. This particular example gives a non-spatial analysis (focusing on a single region with 100,000 individuals), but a list of different areas could be provided (see §3.6 for further details on spatial model). Another column must have the heading "population" (or alternatively several columns stratifying the population into different demographic groups, see §3.5).

'data_tables' incorporates data files (which can again be in ".csv" or ".txt" formats). In the example above, 'data_tables' reads in the number of transitions between compartments E and I per day from the file "EI.csv" (note, when run in inference mode these files should be placed in the data directory, whereas in simulation mode these data files are actually generated in the "Simulate_data" sub-directory in the output). Figure 4 shows this data file.

'state_outputs' can be used to generate additional graphical plots (*e.g.* see Fig.3(d) and 6(d)).

'modification' is used to specify a changes to the model when performing prediction, as discussed in §1.4.3.

---

[24] With the exception of the infection transition, for which the force of infection determines the transition rate.
[25] If the model contains multiple infectious states, setting different values can be used to set their relative infectivity.

Having introduced this simple example, we now systematically go through different aspects of model and data specification to show how more complex scenarios can be dealt with.

3.2 Priors

In the TOML file above all parameters were assumed known except for R_t, which was taken to have a uniform prior. In fact any specified parameter can have a prior distribution associated with it andthis is used when inference is performed. Three types of priors exist:

Univariate priors – These are used for most model parameters and can take any of the following possibilities: "Fixed(.)" to fix the parameter value, "Uniform(.,.)" for a uniform distribution within a range, "Exp(.)" for an exponential distribution with specified mean and "Gamma(.,.)" for a gamma distribution with specified mean and standard deviation.

Dirichlet priors – For branching probabilities it is necessary to apply Dirichlet priors (because these are required to strictly add to one). Three types of Dirichlet prior specification can be made: 1) "Dir(*)" is used to apply an uninformative Dirichlet flat prior, 2) Dir(alpha) applies an informative prior with a specified value for alpha, and 3) "Dir(mean,sd)" applies an informative prior with a specified mean and standard deviation[26].

Modified Dirichlet priors – These prior are used is cases in which the average of a quantity is require to be one, *i.e.* relative susceptibilities, area effects, and level effects[27]. "MDir(k) " applies an informative prior with a specified mean and standard deviation

```
area_effect = { value="[area effect:Scotland_LA.csv]", prior="MDir(0.5)"}
```

Additionally, priors can be set that enforce the ordering of certain parameter. In this example

```
R_spline = [{param="R0 | R1 | R2 | R2", value="2.0 | 1.9 | 1.5 | 0.6", prior="Uniform(0.4,4)"}]
prior_order="R0 > R1 | R1 > R2 | R2 > R3"
```

the command 'prior_order' ensures that under inference the reproduction number R_t must strictly reduce over time. This may be consistent with the prior belief that implementing intervention strategies must inevitably reduce disease transmission.

It is important to note that data often provides little information about some model parameters[28]. If priors are uninformative and diffuse, this can result in a much longer computational time to perform inference than otherwise. Therefore, wherever possible it is advisable to place as restrictive priors on model parameters

---

[26] For *N* Dirichlet parameters, *N*-1 means are specified and a single value for 'sd' (all other values are set to '*'). This restriction arises because the Dirichlet prior support only *N* free parameters.
[27] This ensures the normalisation conditions in Eq.(3), (5) and (7) are strictly satisfied.
[28] In which case the posterior distribution maps out almost the same distribution as the prior.

3.3 Splines

Piecewise linear splines are used in BEEP to capture time variation in model parameters (see Fig. 10). We now provide a description of how these splines are defined (note, the examples below use R_t, but the same basic definitions apply to all splines).

3.3.1 Simulation

First we look at defining a spline for use in simulation. Here is an example (shown in Fig. 10(a)):

> R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="R0 | R1 | R2 | R3",
>   value="2.0 | 1.5 | 0.5 | 1.0"}]

'bp' defines the breakpoints in time along the spline. Each breakpoint is separated by a '|' character (spaces either side of this are ignored). The special keywords 'start' and 'end' refer to the start and end times specified in the analysis. If 'bp' is not set, its default value is taken to be "start|end", *i.e.* representing a simple line across the analysis time period. In the case above, intermediate breakpoints are set on March 1st and July 15th 2020.

'param' defines the names of parameters associated with each of the breakpoints. If 'param' is not specified, BEEP automatically generates informative names for the parameters, *e.g.* "R(t) t:0" would be the name given to the first parameter along this spline.

'value' sets the values those parameters take under simulation.

.
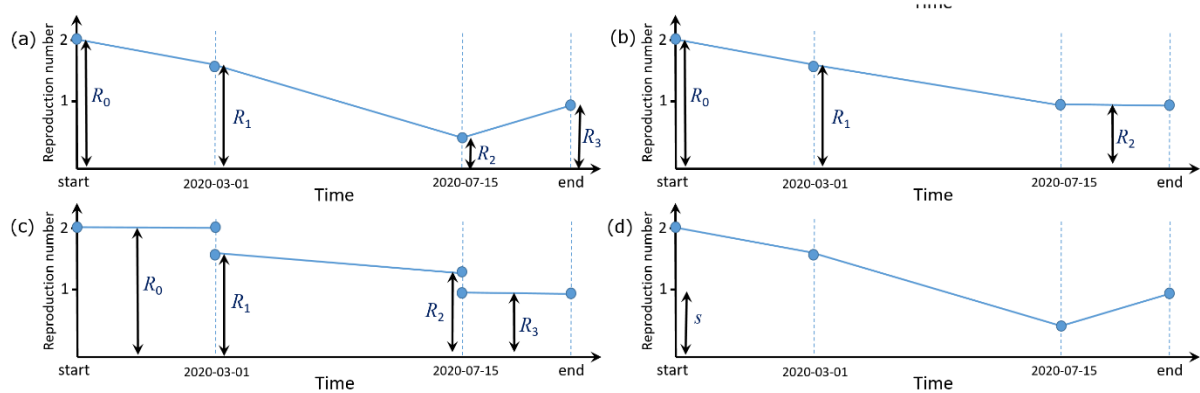
Figure 10: Types of spline – This shows different ways in which splines can be defined (see §3.3 for details). (a) A piecewise linear spline defined by four breakpoints with four parameters R_0-R_3 specify the heights at each breakpoint. (b) The last two breakpoints share the same parameter (hence the spline is flat during this region). (c) This spline has discontinuities. (d) This spline uses a time-varying factor to specify it (a single parameter multiplies the entire profile).

3.3.2 Inference

When performing inference the expression above could still be used, but this would fix the spline to a specified set of parameters. In most cases we actually wish to estimate these parameters from the data. To do this we must add in a specification for the prior:

> R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="R0 | R1 | R2 | R3",

```
                value="2.0 | 1.5 | 0.5 | 1.0", prior="Uniform(0.4,4)"}]
```

By default, the same prior specification gets applied to all spline parameters, although individual specification is also possible. For example

```
  prior="Uniform(0.4,4) | Exp(1.1) | Uniform(0.2,2) | Uniform(1.0,3)"
```

sets a different priors for each parameter.

As well as a prior on each of the model parameters, a smoothing prior can also be placed on the splines itself. This is used to suppress unphysical fluctuations. For example during a real pandemic we would not expect the reproduction number to go drastically up or down week on week. Rather we would expect it to vary in a smooth way. This is incorporated into the model by the addition of:

```
  smooth_type="smooth", smooth = "0.5"
```

'smooth_type' specifies the type of smoothing and 'smooth' sets its strength s[29]. Three different types of smoothing can be selected from:

- "no_smooth" – Implies no smoothing (used as default).
- "smooth" – Implies a smoothing prior given by

$$\pi(\theta^{spline}) = \prod_{b=1}^{B-1} N(\theta_{b+1}^{spline} \mid \theta_b^{spline}, s^2), \tag{28}$$

  where N is the normal probability distribution, b indexes the B breakpoints and θ_b^{spline} represents the parameters along on the breakpoints.

- "log_smooth" – Implies a smoothing prior given by

$$\pi(\theta^{spline}) = \prod_{b=1}^{B-1} LN(\theta_{b+1}^{spline} \mid \theta_b^{spline}, s^2), \tag{29}$$

  where LN is the log-normal probability distribution in this case. This type of smoothing is appropriate when the spline represents a strictly positive quantity.

3.3.3 Different types of spline

Figure 10 shows some different ways in which splines can be defined:

Parameter repeated – Figure 10(b) shows an example in which a single parameter is used on more than one breakpoint[30]:

```
  R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="R0 | R1 | R2 | R2",
                value="2.0 | 1.5 | 1.0 | 1.0"}]
```

Discontinuity – Figure 10(c) shows how discontinuities can be added by repeating the same breakpoint times:

```
  R_spline = [{bp="start | 2020-03-01 | 2020-03-01 | 2020-07-15 | 2020-07-15  | end",
```

---

[29] This strength can be set individually for each breakpoint by separating values using the '|' character.
[30] In this case, the numbers set in 'values' must be the same each time the same parameter is used.

```
                param="R0 | R0 | R1 | R2 | R3 | R3", value="2.0 | 2.0 | 1.6 | 1.4 | 1.0 | 1.0"}]
```

Note, these discontinuities are not acted on by smoothing priors in Eqs.(28) and (29).

Time-varying factor – Figure 10(d) shows an example in which a time-varying factor is used. Here the shape of the curve is fixed, but just a single parameter s multiplies the entire curve (hence during inference only a single parameter needs to be estimated):

```
  R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="s",
              value= "1.0", factor="2.0 | 1.5 | 0.5 | 1.0"}]
```

Whilst not usually appropriate for reproduction number, such an approach can be applicable to the external force of infection (for which movement data between the population under study and elsewhere can potentially be estimated, *e.g.* by air traffic movement).

3.3.4 Using a file to define a spline

If the spline contains numerous breakpoints, definition of them in the TOML file itself can be somewhat cumbersome. Instead, BEEP allows for the possibility of loading information about the spline from a file. For example, the spline used in Fig. 10(c) could equally be represented by

```
  R_spline = [{bp="[Breakpoint:R.csv]", param="[Parameter:R.csv]", value="[Value:R.csv]"}]
```

where the file 'R.csv' is shown in Fig. 11 (this uses the format "[Column name:File name]".

3.3.5 Other spline properties

'geo_filt' – As applied to 'R_spline' and 'efoi_spline', this specifies the area on which the spline acts (optional, set to all areas by default).

'name' – Specifies the name of the spline. This is used when the spline is referred to elsewhere in the TOML code.

	A	B	C
1	Breakpoint	Parameter	Value
2	start	R0	2
3	01/03/2020	R0	2
4	01/03/2020	R1	1.6
5	15/07/2020	R2	1.4
6	15/07/2020	R3	1
7	end	R3	1

Figure 11. Spline file. Shows an example file "R.csv" used to define a spline.

3.4 Setting up the compartmental model

In the notation below we take 'X' to represent either 'param' to specify a parameter name (optional), 'value' to specify its numerical value (for simulation or also inference, if that value is fixed) or 'prior' to specify the prior distribution for that parameter (inference only).

3.4.1 Compartments and transitions

Compartments are set up using the 'comp' command. The following example sets up an SEIRD model (in this model individuals can either recover or die after becoming infected):

```
  comps = [{name="S"},
          {name="E", dist="Erlang", k="2", mean_value="3", mean_prior="Uniform(2,4)"},
          {name="I", dist="Exp", mean_value="4", mean_prior="Uniform(3,5)", inf_value="1"},
          {name="R"},
          {name="D"}]
```

'name' gives the name for the compartment.

'dist' gives the distribution type (with shape parameter k specified for Erlang distributions).

'mean_X' defines the mean residency time (appropriate if transitions leave the compartment). This can be set separately for different demographic groups (see §3.4.2). In this example, a uniform prior has been placed on of potential residency times within compartments E and I.

'inf_X' defines the relative infectivity (assumed zero if unspecified).

Transitions in the model are setup using the command 'trans', for example

```
trans = [{from="S", to="E", infection="yes"},
         {from="E", to="I"},
         {from="I", to="R", prob_value="0.9", prob_prior="Dir(*)"},
         {from="I", to="D", prob_value="*", prob_prior="Dir(*)"}]
```

'from' and 'to' specify the initial and final compartments of the transition.

'infection' specifies the infection transition (only one may be set).

'prob_X' specifies the probability of moving down a given branch (note, this only needs to be specified when the 'from' compartment has more than one transition leaving it). Because the sum of branching probabilities for all transitions leaving a compartment must be one, so one of these probabilities does not need to be specified (because it can be calculated from the others). This is indicated by using the '*' symbol. Here 'prob_prior' defines a flat Dirichlet prior (see §3.2).

3.4.2 Demographic dependency

It might be expected that the mean residency times and branching probabilities, introduced in the previous section, depend on the demographic grouping to which an individual belongs (see §3.5 for how these grouping are defined).

For example, suppose a model contains three age groups: "child", "adult" and "elderly". Within 'comp', an age-based dependency on the recovery time could be added in the following way:

```
{from="I", to="R", mean_param="IR_Y | IR_A | IR_E", mean_value="age: 2.6 | 3.4 | 5"}
```

Within either 'mean_param' or 'mean_value' the specification of demographic classification followed by a colon sets the demographic dependency. Note, the order of the values in 'mean_value' must follow the order in which the demographic categories are defined within the classification (see §3.5).

If there is a further dependency on another demographic classification, this can also be set. For example

```
{from="I", to="R", mean_param="tE_YM | tE_AM | tE_EM | tE_YF | tE_AF | tE_EF",
                   mean_value="age,Sex : 3.4 | 4 | 2.3 | 3.6 | 4.2 | 5.5"
```

would be used to specify age and sex dependency.

Similarly, demographic dependencies can be added to branching probabilities in 'trans'.

3.4.3 Specifying the reproduction number R_t

The command 'R_spline' is used to specify R_t, a spline giving time variation in the reproduction number in Eq.(1) . For example,

```
R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="R0 | R1 | R2 | R3",
             value="2.0 | 1.5 | 0.5 | 1.0", prior="Uniform(0.4,4)"}]
```

sets up the spline shown Fig. 10(a), with a prior restricting its value to the range 0.4 to 4 under inference (see §3.3 for further details on spline specification).

In spatial models the same spline R_t is typically used for all the areas in the system. BEEP does, however, allow for different splines to be fit to different areas (or aggregations of areas) separately using 'geo_filt'[31]. In this example, a different spline is fit to England and Scotland[32]:

```
R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="EngR0|EngR1|EngR2|EngR3",
             value="2.0 | 1.5 | 0.5 | 1.0", prior="Uniform(0.4,4)", geo_filt="Nation:England"},
            {bp="start | 2020-03-01 | 2020-07-15 | end", param="ScoR0|ScoR1|ScoR2|ScoR3",
             value="2.3 | 1.2 | 1.3 | 0.9", prior="Uniform(0.4,4)", geo_filt="Nation:Scotland"}]
```

3.4.4 Specifying the external force of infection

The command 'efoi_spline' is used to define a spline setting the external force of infection $\eta_{s,a,d,t}$ in Eq.(1). This term generates infections in the system initiated elsewhere.

Unlike R_t, it is usually not possible to infer the shape of the profile of external infection[33]. However one way in which it can be incorporate is the following:

```
efoi_spline = [{param="phi", value="1", bp="[bp:phi_spline.txt]", factor="[fac:phi_spline.txt]"}]
```

Here the 'factor' property is used to specify the daily expected external force of infection (which can be derived from, *e.g.*, movement in and out of the area and rates of disease in the interacting population).

Other optional properties can be set: 1) As with R_t above, 'geo_filt' can be set to specify different geographical regions separately, 2) 'strain' can be used to specify a force of infection for a specific strain (see §3.8), and 3) 'age_dist' can set the fraction of externally infected individuals in different age groups.

3.5 Defining demographic groups

It may be that, from an epidemiological point of view, different demographic groups in the population behave differently. This section described how these groupings are specified.

---

[31] In essence R_t is replaced by $R_{a,t}$ in Eq.(1).
[32] Note, from section 3.6, it is assumed that the 'areas' file contains a column with heading "Nation" and either "England" or "Scotland" is specified on each of the rows.
[33] This is because once local transmission gets going the external contribution to the overall infection rate is typically very low.

3.5.1 Age stratification

Age stratification in the population can be incorporated by defining 'ages', for example

```
ages = {cats="age0-19 | age20-69 | age70+"}
```

specifies three age groups (young, adult and elderly).

'cats' specifies the categories (separated by '|'). How the population is divided into age groups is discussed in §3.6.2.

If 'ages' has N age categories, 'age_mixing_matrix' must be set to a file giving an NxN matrix that captures interactions between individuals in the same and different age groups (this is used to derive the matrix $C_{z,z'}$ just below Eq.(9), as explained in Appendix D).

On top of this BEEP allow for modification of $C_{z,z'}$ through the 'age_mixing_modify' command. For example

```
age_mixing_modify = [{type="row_column", agecat="age70+", bp="[bp:amm.csv]",
        value="[value:amm.csv]", prior="Uniform(0,5)", smooth_type="log_smooth", smooth="0.3"}]
```

sets up a smoothed spline which multiplies the row and column of $C_{z,z'}$ corresponding to the age category "age70+". For example, this could be used to study the effect of varying levels of shielding for the elderly.

3.5.2 Other demographic classifications

Other stratifications can be specified by setting 'democats', for example

```
democats = [{name="Sex", cats="Male | Female"},
            {name="Ethnicity", cats="White | Black | Asian"}]
```

'name' describes the classification and 'cats' provides the different categories that classification can take (separated by '|'). Note, the CPU time of the algorithm scales multiplicatively with the classification sizes (hence this model would run approximately 2×3=6 times slower than a non-stratified model).

3.5.3 Variation in susceptibility

Optionally, differences in susceptibility for different age and/or demographic groups can be set or inferred (these corresponds to the values of $\hat{\sigma}_{q,v}$ in Eq.(2)). For example

```
ages = { cats="age0-19 | age20-69 | age70+", sus_value="0.7 | 1.0 | *", sus_prior="MDir(0.5)"}
```

'sus_X' defines the relative susceptibility for different age groups (optional, set to 1 by default). Note, because of the normalisation condition in Eq.(3), not all values need to specified, which is why one value is set to the '*' symbol. 'sus_prior' must take a Dirichlet distribution.

3.5.4 Incorporating changes in demographic groups

The command 'democat_change' allows for the proportions within a specified demographic classification in the susceptible population to vary over time (*e.g.* this can be used to study the effect of vaccination), for example

```
democats = [{name="Vac Status", cats="Vac | Not Vac"}]
democat_change = [{name="Vac Status", file="vac.csv"}]
```

'name' references the name for the demographic classification that is changing.

'file' gives the name of the data file (see Fig. 12 as an example). This must have a 'Date' column and columns for all but one of the demographic categories. The numbers can either be populations, percentages or fractions. At times between the specified dates the demographic make-up is linearly interpolated.

Two further specifications can be made:

'democats_filt' specifies a filter to include only a specified demographic groups (*e.g.* 'democats_filt ="Sex:Male"' would mean that the specified demographic changes are made on males only).

'geo_filt' specifies a geography over which the observation are made (*e.g.* 'geo_filt ="Nation:Scotland"' would mean that specified demographic changes are made in Scotland only.

	A	B
1	Date	Vac
2	2021-01-01	0
3	2021-02-01	75400
4	2021-03-01	143455
5	2021-04-01	284530
6	2021-05-01	545356
7	2021-06-01	843564

Figure 12. Vaccination data. Shows example file "vac.csv".

	A	B	C	D	E	F	G	H	I	J
1	area	name	nhs region	density	age0-19	age20-69	age70+	Male	Female	area effect
2	S12000005	Clackmannanshire	S08000019	3.23327	11269	33087	7044	25225	26175	1.3
3	S12000006	Dumfries and Galloway	S08000017	0.231554	29158	92583	27049	72321	76469	0.7
4	S12000008	East Ayrshire	S08000015	0.965354	26223	78457	17160	59105	62735	1.2
5	S12000010	East Lothian	S08000024	1.557614	23750	66839	15201	50719	55071	0.6
6	S12000011	East Renfrewshire	S08000031	5.461709	23834	57594	13742	45473	49697	1.5
7	S12000013	Na h-Eileanan Siar	S08000028	0.0878	5305	16648	4877	13247	13583	2.3
8	S12000014	Falkirk	S08000019	5.392018	35007	104352	20981	78497	81843	0.6
9	S12000017	Highland	S08000022	0.091814	49231	149837	36472	115391	120149	0.9
10	S12000018	Inverclyde	S08000031	4.871019	15906	50622	11622	37401	40749	1.3

Figure 13: Area specification. This gives an example of a file specifying different local authorities in Scotland (taken from "example/Data_EX2/Scotland_LA.csv"). The columns are as follows: 'area' specifies a unique area code, 'name' gives the name of the local authority (not used in analysis), 'nhs region' specifies which NHS region the local authority belongs to, 'density' gives the population density, 'age#' gives populations in different age groups, 'Male' and 'Female' give populations broken into sexes, and, finally, 'area effect' is a fictional relative area effect data used for simulation purposes.

3.6 Spatial models

3.6.1 Loading area information

The 'areas' command is used to specify a file in the data directory giving information about the population under study and the geographical region in which that population resides, for example

```
areas= "Scotland_LA.csv"
```

Figure 13 shows an extract of a file giving information about local authorities in Scotland.

Generally speaking, the 'areas' file must contains the following columns: "area" (which gives a code or name that uniquely identifies each area), other coarser geographical scales if they are used (*e.g.* 'nhs region' amalgamates local authorities into NHS regions), columns for covariates in the model (*e.g.* population density)[34] and, potentially, the initial population make-up (see below).

3.6.2 The initial population

This section discusses how information about the initial population ($p_{a,d_s,c,,t_{start}}$ in §2.1.3) is loaded into BEEP. In fact, it can be incorporated in one of two ways:

Within the 'areas' file – If the model is not age-stratified (*i.e.* the command 'ages' is unspecified), a column with the heading "population" in the file 'areas' gives the total population size within each area. For age-stratified models, in the first instance BEEP looks for column headings in the 'areas' file corresponding to the specified age categories (*e.g.* "age0-19", see Fig. 13). Failing that, it takes the text preceding the number range (which is "age" is this particular case), and then looks to add up columns with headings given by this text plus a number in the range specified[35]. For other demographic classifications the 'areas' file must contain column headings corresponding to each category, with values given as either populations, percentages, or fractions. Here we assume that probabilities of being in different categorisations are independent, *e.g.* the population of males in the age range 0-19 would be equal to the population in 0-19 multiplied the fraction of males. This assumption is usually reasonably valid in most cases, although it can break down (*e.g.* individuals in older age groups are predominantly female). To overcome this limitation the following option can be used.

	A	B	C	D	E
1	area	age	Sex	compartment	population
2	S12000005	age0-19	Male	S	5530
3	S12000005	age20-69	Male	S	16238
4	S12000005	age70+	Male	S	3457
5	S12000005	age0-19	Female	S	5739
6	S12000005	age20-69	Female	S	16849
7	S12000005	age70+	Female	S	3587
8	S12000006	age0-19	Male	S	14173
9	S12000006	age20-69	Male	S	45001
10	S12000006	age70+	Male	S	13147

Figure 14. Population initialisation file. This shows an example of a file that could be used with 'init_pop' to specify the initial population. Note, the ordering of row/columns is unimportant.

Using initialisation file 'init_pop' – Alternatively the initial populations can be specified within a file specified using the 'init_pop' command. This file must contain the headings "area" (giving the area name, as specified in the 'areas' file), "age" (if the model is age stratified), the names for any demographic categories, "strain" (if different strains are specified), "compartment" (specifying the name of the compartment) and, finally, "population". The file then consists of series of rows that list every possible combination of options followed by the corresponding population within that grouping (*e.g.* see Fig. 14).

---

[34] The columns do not have to have any particular order, provided they have the correct headings. Other columns may also exist which are not used by the analysis.

[35] Suppose 'areas' contained columns for yearly age intervals: "age0", "age1",....,"age99". An age category set to "age0-19" would correctly add up all columns from "age0" and "age19" and use this for analysis. Additionally, ending with a + sign, *e.g.* "age70+", adds up all columns at and above that value.

3.6.3 Geographical mixing

When more than one area is specified it becomes necessary to load a matrix that captures the interactions between areas. This is achieved by the following command

```
geo_mixing_matrix = "census_flow_data.csv"
```

If the model contains A areas, the specified file must contain an $A{\times}A$ table (with no headings) that sets $Z_{a,a'}$ in Eq.(8). This matrix is proportional to the rate at which an individual in one area (in the y-direction of the matrix) comes into effective contact with an individual in another area (in the x-direction of the matrix). Note, this differs from how the age mixing matrix is defined and it ensures that $Z_{a,a'}$ is symmetric. A description on how to derive such a matrix from census flow data is provided in Appendix C. $Z_{a,a'}$ only needs to be specified up to a constant factor.

In addition, the spline m_t in Eq.(8), which allows for time0variation in mixing, is specified by the command 'geo_mixing_modify'. For example

```
geo_mixing_modify = [{bp="[bp:gmm.csv]", value="[value:gmm.csv]", prior="Uniform(0,1)"}]
```

sets a spline which is defined using the file 'gmm.csv' and under inference is bounded between 0 and 1 (see §3.3 for more details on spline). If not specified, m_t=1 is assumed.

3.6.4 Area effects

The following four commands refer to the four contributions to area effects in Eq.(4).

Firstly, relative area effects w_a are incorporated by

```
area_effect = {value="[area effect:Scotland_LA.csv]", prior="MDir(0.5)"}
```

Note, in Fig. 13 the column 'area effect' contains values used when simulating from the model, and a flat Dirichlet prior is used when performing inference (this ensures the average area effect is 1).

Secondly, fixed effects. Columns of the design matrix X are added using

```
area_covars= [{name="density", param="b1", value="0.1", prior="Uniform(0,1)", func="log"}]
```

'name' must correspond to one of the columns in the 'areas' file (see Fig. 13).

'func' gives an optional functional transformation given to the raw data (this can be "linear" to leave it unchanged, or "log" to log transform).

This example shows one fixed effect but more columns to X can be added within the square brackets.

Thirdly, time-varying fixed effects. These are factors which might, in principle, affect disease transition, *e.g.* temperature or rainfall. For example

```
area_tv_covars = [{ name="Tarea", file="areatvcovar.csv", param="b2", value="0.1",
                    prior="Uniform(0,1)", func="linear"}]
```

where the file 'tvcovars_spatial.csv' contains information about the matrix $X'_{a,t}$ for a single fixed effect. 'tvcovars_spatial.csv' must contain time-varying columns for each of the areas (see Fig. 15 for an example).

In cases in which $X'_{a,t}$ is independent of area a, the following command can be used:

```
tv_covars = [{name="covar", file="tvcovars.csv", param="b2", value="0.1",
                prior="Uniform(0,1)", func="linear"}]
```

where the file 'tvcovars.csv' would contain a column specified in 'name' that would inform X'_t.

Lastly, level effects allow for different levels of disease intervention to be estimated. The level effects matrix $l_{a,t}$ is loaded using

```
level_effect = {file="level.csv", param="level1 | level2", value="0.5 | *", prior="MDir(0.5)"
```

The file 'level.csv' contains a table with dates going down rows and areas along columns, as shown in Fig 16.

Figure 15: For time-varying fixed effects. This shows an extract from 'examples/Data_EX2/areatvcovars.csv', which is a fictional dataset giving average air temperature. Different columns represent areas (see Fig. 13 for area specification) and different rows give dates.

	A	B	C	D	E	F	G	H	I	J	K
1	Date	S12000005	S12000006	S12000008	S12000010	S12000011	S12000013	S12000014	S12000017	S12000018	S12000019
2	2020-01-01	level1	level1	level2	level1	level1	level2	level2	level2	level2	level2
3	2020-01-02	level1	level1	level2	level1	level1	level2	level2	level2	level2	level2
4	2020-01-03	level2	level1	level2	level1	level1	level2	level2	level2	level2	level2
5	2020-01-04	level2	level1	level2	level1	level1	level2	level2	level2	level2	level2
6	2020-01-05	level2	level1	level2	level1	level1	level2	level2	level2	level2	level2
7	2020-01-06	level2	level1	level2	level1	level1	level2	level2	level2	level2	level2
8	2020-01-07	level1	level1	level2	level1	level1	level2	level2	level2	level2	level2
9	2020-01-08	level1	level1	level2	level1	level2	level2	level2	level2	level2	level2
10	2020-01-09	level1	level1	level2	level1	level2	level2	level2	level2	level2	level2

Figure 16: Level effects. This shows an extract from the file 'examples/Data_EX2/level.csv'. This fictional dataset assigns each area to one of two disease intervention levels as a function of time. Different columns represent areas (see Fig. 13 for area specification) and different rows give dates.

3.7 Data tables

This section describes how the command 'data_tables' is used to load the different types of data discussed in §2.2. For example

```
data_tables = [{type="transition", observation="E->I", timestep="7", file="E-I.csv",
                obsmodel ="normal 10%", shift="2"}]
```

'type' indicates the type of data, which can take four different values:

- "transition" – This gives time series information about the number of individuals moving down a specified transition(s).
- "population" – This gives time series population numbers within a specified compartment)s_.
- "population_fraction" – This gives time series population fractions within a specified compartment(s).
- "marginal" - This gives the total number of transitions over a given time period, typically stratified by a demographic classification.

'observation' describes what is observed. When the type is "transition" or "marginal" this is given by the initial compartment followed by "->" and then the final compartment (additionally, multiple transitions can be specified, *e.g.* "I->D,I->R", and the sum of them is used). For "population" or "population_fraction" this is simply the compartment under measurement (or several compartments that are comma separated).

'timestep' sets an integer which gives the number of time units between observations.

'file' specifies the name of the file. This must be either a tab-separated '.txt' text file or a '.csv' file.

'obsmodel' sets the observation model, which relates the observations to the underlying system state. Five possibilities exist: "normal #%" (where '#' is a number giving the percentage error), "normal #" (where '#' is an absolute standard deviation), "normal load" (where the standard deviation is loaded from a file), "poisson", and "negbin" (see §2.3.4 for details). In the case of "negbin" then "shape" must also be specified. See §2.3.4 for details.

Three further optional specifications can be made regarding timings:

'shift' allows for the date/time of the data to be shifted by a specified number (*e.g.* case data can be used as a proxy for infection times, but it first needs to be shifted to account for the difference in time between an individual becoming infected and when they become a case).

'start' and 'end' gives the first and last date/time for the data. In the case of time series data these are usually just automatically taken from the data tables provided. For marginal data they are assumed, unless specified, to coincide with the analysis period.

3.7.1 Geographical and demographic filtering

Considering transition data, the 'observation' property allows us to specify which transition is observed. However, suppose we actually separately have measurements for each of the areas in the system. This is an example of adding a geographic dependency. Alternatively the data might only provide information about males. This is an example of demographic filtering. The following specifications can be made within 'data_tables' to add geographical or demographic dependencies as well as filters:

'geo_dep' allows for multiple columns in the data file to each refer to a different geographic group, *e.g.* 'geo_dep="area"' would give columns for each of the geographical areas, but other geographies

could be used (if we set 'geo_dep="nhs_region"' then the data would apply to the different NHS regions in Fig. 13).

'geo_filt' filters the geography over which the observation are made, *e.g.* 'geo_filt="nhs_region: S08000019"'' would mean that the data would apply to only that specific NHS region. If more than one option is filtered then they can be comma separated.

'democats_dep' allows for multiple columns in the data file to each refer to a different demographic group, *e.g.* 'democats_dep="Sex"'' would imply that the data has two columns, one for 'Male' and one for 'Female'). For age dependency 'democats_dep="age"'' is used.

'democats_filt' filters the demographic group observed (*e.g.* 'democats_filt="Sex:Male"'' would mean that observations are made on males only).

3.7.2 Missing and thresholded data

In cases in which the data contains missing values the 'nodata_str' TOML command can be set which specifies the character or string that denotes missing value, *e.g.* 'nodata_str = "NA"'' (by default this is set to ".").

For non-identifiability reasons, in some cases data is "thresholded". This means a particular character or string is used to represent the fact that the data is at or below a specified threshold. This string is specified by the 'threshold_str' command (*e.g.* 'threshold_str="<=5"') and the numerical threshold itself is set in 'data_tables' by the 'threshold' property.

3.7.3 Partial observation

BEEP provides two ways to incorporate the fact that raw data may need be transformed before it can be incorporated into the model (as discussed in §2.3.5):

'factor' sets a factor which multiplies the value given in 'observation' to get to the data, *e.g.* this can be used to incorporate a known test sensitivity (optional, by default set to 1).

'factor_spline' references the name of a spline that sets the factor between the value given in 'observation' and the observed data (*i.e.* s_t in Eq.(24)). The spline itself is specified by the 'obs_spline' command. For example

```
data_tables = [{type="transition", observation="E->I", file="IH.csv", factor_spline="obs1"}]
obs_spline = [{name="obs1", value="0.1|0.3|0.5", prior="Uniform(0,1)", bp="start|20.03.20|end"}]
```

implies that the fraction of actual E->I transitions observed varies, starting at 10% and ending at 50%. As well as being able to set the spline s_t, it can also be inferred from the data.

3.8 Different disease strains

Different strains of a disease can have different properties, *e.g.* variation in transmission rate and recovery time. They can be incorporated into the model using the 'strains' command, for example

```
strains = {cats="s1 | s2", sus_value="0.7 | *", sus_prior="MDir(0.5)",
          Rfactor_value="1.0 | 2.0", Rfactor_prior="Fixed(1) | Uniform(0.5,2)"}
```

'cats' provides names for the different strains.

'sus_X' can optionally be used to vary the susceptibility of individuals to different strains.

'Rfactor_X' sets the factor by which R_t is increased to account for a difference in transmission rate (typically one strain is set to 1 to become the reference for the others). This sets the parameter ψ_s in Eq.(1).

In terms of filtering and dependencies in §3.7.1, strains are treated like a demographic classification, *e.g.* 'democats_filt="strain:s1"' would be used to add data about strain 's1'.

3.9 System dynamics

As discussed in §2.1.3, continuous time Gillespie algorithm system dynamics are approximated using a finite time step τ-leaping algorithm. The value τ is set by $1/Q$, where integer Q is the number of discrete time steps per unit time. Q can be set by

```
steps_per_unit_time = 2
```

and by default it has a value 4 (*i.e.* τ is a quarter of a day). Reducing Q can reduce the computational time to run inference, although this can also lead to substantial discretisation error. Crucially, τ should be substantially less than the compartmental residency times specified in the model.

The underlying system dynamics can be set using the command

```
dynamics = "deterministic"
```

or alternatively set to "stochastic" (which is assumed by default)[36].

3.10 Model modification

Modifications to the model (as used when making predictions in §4.3) are specified using the 'modification' command. For example

```
modification = [{type="trans_rate_fac", start="2020-4-01", trans="S->E", factor="0.5"}]
```

specifies that from April 1st the number of infections in the model reduced by a factor of a half (in essence something causes disease transmission to become reduced).

Here we list different possible modifications that can be applied to the model:

'type' denotes the type of change. This can be one of the following:

- "trans_rate_fac" – This changes the rate of a specified transition by a factor ('trans' and 'factor' must be set).
- "beta_fac" – Changes the transmission rate by a factor ('factor' must be set).
- "efoi_fac" – Changes the external force of infection by a factor ('factor' must be set).
- "spline_fac" – Multiplies a spline by a factor ('name' must correspond to 'name' set in a spline and 'factor' must be set).

---

[36] When the system dynamics are deterministic, the ABC-MBP, PAIS-MBP, MCMC-MBP and MC^3 inference algorithms cannot be used as MBPs rely on stochastic dynamics to operate.

- "spline_set" – Sets the value of a spline ('name' must correspond to 'name' set in a spline and 'factor' must be set).

'start' and 'end' gives the duration for the change.

'strain' is specifies if only a particular strain is referred to (optional).

'geo_filt' sets the modification to apply to a particular geography (optional).

'democats_filt' sets the modification to apply to a particular demographic category (optional).

3.11 Tailoring Outputs

3.11.1 Generating state outputs

In the output (see §5), it is possible to add other informative plots, if needed. The command 'state_outputs' works in much the same way as 'data_tables', but rather than reading input files (or generating simulated data) corresponding to a particular observation, here just graphs are generated. In this example

```
state_outputs = [
        {plot_name="Dynamics", type="population", observation="S", line_colour="green"},
        {plot_name="Dynamics", type="population", observation="E", line_colour="yellow"},
        {plot_name="Dynamics", type="population", observation="I", line_colour="red"},
        {plot_name="Dynamics", type="population", observation="R", line_colour="blue"}]
```

'plot_name' sets the name of the final plot. Using this more than once allows for multiple lines to be placed on the same plot.

'line_colour' sets the colour of the line in the final plot. This can take the values 'black', 'red', 'blue', 'green', 'yellow', 'cyan', 'magenta' (and by default an automatic colour scheme is selected).

See Glossary for other specifications.

3.11.2 Probability of reaching a specified compartment

Under certain circumstance it is of interest to find out the overall probability that individuals reach a particular compartment in the model, *e.g.* if set to the dead D compartment this can be used to calculate the infection fatality rate. This can be specified using the 'prob_reach' command, *e.g.*

```
prob_reach = [{name="ifr", comp="D"}]
```

'name' is used in the output file.

'comp' specifies the compartment which is reached.

3.11.3 Probability distribution plots

BEEP allows some flexibility in the way in which probability distributions are plotted, as specified in 'output_prop'. For example

```
output_prop = {probdist="kde", h="5"}
```

'probdist' sets how probability distributions are displayed: "kde" for kernel density estimation or "bin" for binning.

'nbin' determines the number of bins used when binning (optional, 200 by default).

'h' sets a smoothness parameter when using kernel density estimation (optional, 10 by default).

3.11.4 Setting time labels

When setting up the model and generating outputs, it is sometimes useful to represent specific times of interest in the analysis with labels. These can be specified in the following way:

```
time_labels = [{name="Lockdown", time="2020-03-23"}]
```

Further labels can be added by comma separating them inside the square brackets. These labels can subsequent be used instead of dates in the TOML file (*e.g.* when specifying the breakpoints in a spline), and are also placed as vertical reference lines in the final time-based output graphs.

3.11.5 Adding simulated values to graphs

When testing the algorithm it is often useful to simulation from the model using a given set of model parameters, and then perform inference on those parameters to check if they are correctly identified. To place these simulated values onto output plots then add the following command:

```
plot_param_values = "true"
```

3.11.6 Display state curves

By default, state graphs are plotted using posterior means along with 95% credible intervals indicated by dashed lines (*e.g.*, see Fig. 6). An alternative is to explicitly plot the individual posterior samples using faint blue lines (or alternatively plot multiple simulations in "mutisim" mode). To visualise these outputs the following command is used:

```
state_uncertainty = "curves"
```

4 Running BEEP

Once the input TOML file is specified and BEEP is compiled (using the "make" command) it can be run by executing "build/bin/BEEP" on the command line followed by a specification of the input file name. Following that, additional commands can be specified[37] that can be used to quickly change analysis without having to edit the input file each time.

4.1 Simulation-based approaches

Two different simulation-based modes of operation can be selected from:

4.1.1 Single Simulation

This simulates the state of the system given a set of model parameters.

```
build/bin/BEEP inputfile="file.toml" mode="sim" seed=10
```

---

[37] Note, these will override any existing specifications in the TOML file itself.

Since simulations are stochastic, the outcome will depend on how the pseudorandom number generator is initialised. In BEEP this is set by specifying a 'seed'[38] (optional).

4.1.2 Multiple simulation

This simulates from the model multiple times and outputs the distributions in states generated. This make it possible to visualise the level of inherent stochastic variation in the model.

```
build/bin/BEEP inputfile="file.toml" mode="multisim" nsimulation=100
```

This has the following option:

- *nsimulation* – specifies the number of simulations to be performed.

4.2 Inference

BEEP incorporates eight different inference algorithms (briefly discussed in §2.3) that all aim to perform essentially the same task: to generate posterior samples for model parameters and the system state. Which of these algorithms to choose, however, may depend on the scenario under consideration, with some methods performing computationally better or worst depending on the situation. They are split into three broad classes with different aims: 1) maximise an approximation to the *a posteriori* probability directly, 2) reduce an error function (ABC-MBP, ABC, ABC-SMC) and 3) incorporate a specified observation model (PAIS-MBP, PMCMC, MCMC-MBP, MC^3). In most cases ABC-MBP seems to work best for the former and PAIS-MBP for the latter.

As well as the options below, each algorithm can set the 'nrun' option, which replicates analysis over multiple runs. This is normally recommended, as it checks that all runs converge on the same posterior (through Gelman-Rubin diagnostic statistics).

4.2.1 Maximum *a posteriori* probability using CMA-ES

The following command implements the ABC-MBP algorithm:

```
build/bin/BEEP inputfile="file.toml"  mode="map"
```

with the following options:

- *nparticle* – Sets the number of particles, by default set to 4+(5*log(nvar)) rounded to the nearest integer.
- *ngeneration* – Gives the number of generations (by default generations are iterated until there is no substantial change in the MAP estimate for the previous 10 generations).

4.2.2 Approximate Bayesian Computation with model-based proposals (ABC-MBP)

The following command implements the ABC-MBP algorithm:

```
build/bin/BEEP inputfile="file.toml"  mode="abcmbp"  nparticle=200  ngeneration=40
```

with the following options:

---

[38] The 'seed' command can also be set in other modes to test for the influence of this on the outcome.

- *nparticle* – Sets the number of particles (200 is typical as this yields approximately 200 random samples from the posterior).
- *ngeneration* – Gives the number of generations (a suitable value is problem dependent, but typically this lies in the range 10-100, depending on model complexity).
- *cutoff_final* – As an alternative to setting '*ngeneration*', this iterates generations until a specified EF_{cutoff} is reached.
- *nupdate* – Sets the number of MCMC 'updates' used per particle per generation (by default set to 1, but larger values can improving mixing and avoid degeneracy). Each update consists of a variety of MBPs on each of the model parameters as well as on combinations of them.

4.2.3 Particle annealed importance sampling with model-based proposals (PAIS-MBP)

Whereas ABC-MBP aims to generate the posterior by reducing the error function below a cut-off value, here we assume a full observation model and aim to generate the posterior from that.

```
build/bin/BEEP inputfile="file.toml"  mode="pais"  nparticle=200  ngeneration=40
```

This has the same command options as ABC-MBP, except:

- *invT_final* – As an alternative to setting '*ngeneration*' this iterates generations until a specified inverse temperature ϕ_{post} is reached (instead of 'cutoff_final' in ABC-MBP).
- *quench_factor* - Sets the rate of quenching (optional, set to 0.5 by default such that on average half the particle each generation are discarded).

4.2.4 Approximate Bayesian computation (ABC)

This generates posterior samples by means of a simple ABC rejection-sampling scheme.

```
build/bin/BEEP inputfile="file.toml"  mode="abc"  nsample=200 cutoff_frac=0.01
```

with the following options:

- *nsample* – This specifies the number of posterior sample we wish to obtain.
- *cutoff* – Sets the EF_{cutoff} such that only simulated states with EF below this threshold are accepted.
- *cutoff_frac* – An alternative to '*cutoff*', this first generates random samples and then automatically adjusts the EF_{cutoff} such that a specified fraction of them are accepted. This has the advantage that EF_{cutoff} does not need to be chosen (which is difficult in practice), but has the disadvantage of using much more memory (because it is necessary to store all generated states).

4.2.5 Approximate Bayesian computation with sequential Monte Carlo (ABC-SMC)

This runs over a number of generations, with the previous generation used as an importance sampler for the next. Under some circumstances this approach is significantly more computationally efficient than the standard ABC algorithm.

```
build/bin/BEEP inputfile="file.toml"  mode="abcsmc"  nsample=200 ngeneration=5 cutoff_frac=0.5
```

with the following options:

- *nsample* – This specifies the number of samples in each generation.
- *ngeneration* – Gives the number of generations.
- *cutoff_final* – As an alternative to setting '*ngeneration*', this iterates generations until a specified EF_{cutoff} is reached.
- *cutoff_frac* – Specifies how the cut-off in EF is chosen such that a certain fraction of particles are accepted for the next generation (optional, by default set to 0.5).
- prop_size – Sets the proposal size (optional, set to 1 by default). This factor scales the MVN estimate of the posterior distribution in the previous generation to set the parameter proposal kernel.

4.2.6 Particle Markov chain Monte Carlo (PMCMC)

This runs an MCMC chain in parameter space with a particle filter used to generate an unbiased estimate for the agreement between the model and the data.

```
build/bin/BEEP inputfile="file.toml"  mode="pmcmc"  nsample=5000  nparticle=200
```

with the following options:

- nsample – This gives the number of samples on the MCMC chain (note, this is typically much higher than for other methods because successive samples are highly correlated).
- *nparticle* – This gives the number of particles used in the filter (a higher number allows for a better agreement with the data but is computationally slower).
- *invT* – Sets the inverse temperature of the posterior ϕ_{post}.
- *nburnin* – Sets the number of MCMC iteration for burn-in (optional, by default set to a quarter of '*nsample*').
- *nthin* – This thins stored posterior samples by a factor to save memory (optional, by default set to one thousandth of the number of samples).

4.2.7 Markov chain Monte Carlo (MCMC-MBP)

This runs a single MCMC chains and makes use of MBPs for fast mixing.

```
build/bin/BEEP inputfile="file.toml"  mode="mcmcmbp"  nsample=1000  invT=1
```

with the following options:

- *nsample* – This gives the number of samples on the MCMC chain (note this is typically much higher than other methods because successive samples are highly correlated).
- *invT* – Sets the inverse temperature of the posterior ϕ_{post}.
- *nburnin* – Sets the number of MCMC iteration for burn-in (optional, by default set to a quarter of '*nsample*').
- *nthin* – This thins stored posterior samples by a factor to save memory (optional, by default set to one thousandth of the number of samples).

4.2.8 Metropolis-coupled Markov chain Monte Carlo (MC³)

This runs multiple MCMC chains in parallel, with the "hottest" chain mapping out the prior and the "coldest" chain mapping out an approximation to the posterior with a specified inverse temperature.

```
build/bin/BEEP inputfile="file.toml"  mode="mc3"  nsample=1000 nchain=80 invT_final=300
```

with the following options:

- *nchain* – gives the number of MCMC chains used (this needs to be sufficiently large to allow for adjacent chains to "overlap").
- *nsample* – This gives the number of samples on the MCMC chain (note this is typically much higher than other methods because successive samples are highly correlated).
- *invT_final* – Sets the inverse temperature of the posterior chain ϕ_{post}.
- *invT_start* – Sets the inverse temperature for the highest temperature chain (optional, by default set to zero, corresponding to the prior).
- *nburnin* – Sets the number of MCMC iteration for burn-in (optional, by default set to a quarter of '*nsample*').
- *nquench* – Sets the rate at which temperature is quenched during the burn-in period (optional, by default set to half of '*nburnin*').
- *nthin* – This thins stored posterior samples by a factor to save memory (optional, by default set to one thousandth of the number of samples).

4.3 Prediction

When inference is performed a series of posterior samples for model parameters θ and system state ξ are generated and saved into the "Posterior/samples" directory. When run in "prediction" mode, BEEP loads these samples and uses them to generate predictions, for example

```
build/bin/BEEP inputfile="file.toml"  mode="prediction"  prediction_end="2022-4-01"
```

with the following options:

- *prediction_start* – Sets the start time (optional, by default set to either when modification is made to the model or the inference ending time, whichever is earlier).
- *prediction_end* – Sets the end time (optional, by default set to inference ending time).
- *modification* – Allows for specification of any modifications to the model (optional, see §3.10 for details).
- *nsim_per_sample* – Sets the number of simulations per posterior sample (optional, by default set to 4).

Four types of analysis can be performed using "prediction" mode:

Future prediction – Here the aim is take all available data and predict which is likely to happen in the future (assuming transmission rates and other splines remain constant after the end of the period used during inference). This is achieved by setting some future value for 'prediction_end' and viewing the output to see how the system evolves. Alternatively, different future scenarios could be looked at by adding changes to the model through 'modification'.

Counterfactual analysis – Here the aim is to investigate how things would have turned out differently had different measures been taken (*e.g.* alternative disease intervention strategies). This would typically use the same time period as for inference, but make changes to the model through 'modification'.

Posterior predictive checks – These are used to check the model is performing in accordance with the data. Typically a time 'prediction_start' would be selected and then subsequently states generate from simulation would be checked against the observed data. A poor fit implies the model is not good.

4.4 Parallelisation

To increase computational speed BEEP can be run on multiple CPU cores. In this case execution on the command line is preceded by "mpirun –n" followed by the number of cores. For example the ABC-MBP method can be run using:

```
mpirun –n 10 build/bin/BEEP inputfile="file.toml"  mode="abcmbp"  nparticle=200
ngeneration=40
```

This will complete almost 10 times faster because it is run on 10 CPU cores in parallel (which communicate with one other using MPI). Note, when run in parallel the number of particles must be a multiple of the number of CPU cores (so they can be distributed evenly across cores).

5 Outputs from BEEP

5.1 Files generated

We give a brief description below of the various files and folders generated in the output directory once analysis is complete:

- **Parameter_estimates.csv** – If inference is performed, this file gives the posterior means and 95% credible intervals for each of the model parameters. Furthermore, estimates for some derived quantities are also included, *e.g.* the generation time (see Appendix A), and quantities specified in 'prob_reach' (see §3.11.2).
  To check inference has been performed correctly, the estimates for effective samples size[39] for each of the model parameters should exceed around 200 and the Gelman-Rubin statistics should be less than around 1.05.
- **visBEEP.pdf** – This visualises outputs from BEEP.
- **Model_specification.txt –** This gives a summary of the model used to perform the analysis.
- **Posterior** – This folder contains files giving posterior probability distributions for the model parameters. These are arranged in a number of different ways:
  - *parameter* - Contains files for the model parameters.
  - *state* - Contains files which compare the system state with that observed in the actual data files.
  - *spline* - Contains files giving time variation in splines used within the model.
  - *susceptibility* - Contains files giving the variation in susceptibility for different demographic classifications within the model.
  - *sample* - Contains files giving raw posterior samples.
  - *Rmap.csv* - For spatial models this gives the variation in reproduction number R across different areas.

---

[39] Note, not all algorithms generate an estimate for effective sample size.

- **Simulated_data** – This gives simulated data files corresponding to the specifications provided in the input TOML file.
- **Diagnostics** – This provides diagnostic information to inform how well the algorithm is performing:
  - *Generation.csv* - Shows how model parameters move from the prior distribution to an approximation of the posterior distribution as a function of the generation number.
  - *MCMC_proposals.csv* - This shows the performance of MCMC proposals (if used).

5.2 Graphical interface

To aid easy visualisation of results BEEP includes a simple graphical interface which will work in any web browser. This can be viewed by clicking on "visBEEP.html" in the analysis output directory.

Description of the analysis can be added by using

    description = "This analysis uses the following model and the outputs show…"

5.2.1 Maps

When plotting maps the command 'area_plot' can be used to specify a file giving the boundaries of the areas in the system, *e.g.*

    area_plot = { boundary="Scotland_areas.geojson", projection="equirectangular"}

Here '*projection*' is a used to specify how the coordinates are projected onto the map ("equirectangular" is appropriate when longitude/latitude are being used, otherwise "uniform" is used by default). 'area_plot' supports the '.geojson' and '.kml' file formats, and allows for both longitude/latitude measurements as well as grids coordinates. Alternatively 'area_plot' can be used to specify columns giving the average positions of the areas, in which case BEEP will automatically construct a Voronoi tessellation to derive areas from these.

6 Examples

This section goes through a series of examples that help to illustrate the different features of BEEP. Each of these examples has an associated TOML file in the "examples" directory (*e.g.* "EX_A1.toml") as well as a data directory (*e.g.* "Data_EX_A1").

To simulate from one of the model examples use:

    build/bin/BEEP inputfile="examples/EX_#.toml" mode="sim"

where "#" is set to the relevant example, *e.g.* "A1".

Once run, the outputs are placed into the "examples/Output_EX_#" directory. These outputs can be visualised by opening the "visBEEP.html" file (which will work in any modern web-browser). Under simulation, data files specified in 'data_tables' are generated in the "Simulated_data" directory.

To perform inference on of the model /data examples use:

    mpirun -n 20 build/bin/BEEP inputfile="examples/EX_#.toml" mode="map"

Note, this uses the "map" inference methodology, but other algorithms can also be applied (*e.g.* see the top of the "EX_A1.toml" file for different possibilities).

EX: Exemplar used in section 1.4

Objective: Used in the five minute guide to illustrate the broad features of BEEP in a simple example.

Implementation: This model is implemented in the "example/EX_Simple.toml" file (as shown in section 3.1).

Model: An SEIR model is used which divides the population into S: Susceptible, E: Exposed (infected but not infectious), I: Infected (infectious) and R: Recovered. The reproduction number R_t measures the number of secondary infections from an initially infected individual, in an otherwise susceptible population (note, the effective reproduction number R_t^{eff} also accounts for the reduction in the susceptible population over time, and this is a model output). Consequently, the value of R_t is proportional to the rate at which individuals come into contact (with each contact allowing for the possibility of disease transmission). This model assumes that R_t remains constant, and hence the contact rate between individuals remains the same over time. Furthermore, a constant external force of infection is assumed.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number and latent and infectious periods.

A) Introductory examples

EX A1: SIR model

Objective: Introduce a simple epidemiological model to illustrate simulation and inference.

Implementation: This model is implemented in the "example/EX_A1.toml" file. Here 'comps' defines the compartments in the model and 'trans' the transitions. 'R_spline' and 'efoi_spline' define the time variation in reproduction number R_t and external force of infection η_t (both of which are assumed to remain fixed in this example). 'areas' references a file giving information about the overall population (in this case 10,000 individuals intermixing in a single area). 'data_tables' references files giving the weekly number of infections and recoveries.

Model: This model contains three compartments: susceptible (S), infected (I) and recovered (R). Together they are known as the "SIR model". This model assumes that the reproduction number R_t remains constant, and hence the contact rate between individuals remains the same over time. Furthermore, a constant external force of infection is assumed.

Data: The weekly number of S→I infections and I→R recoveries.

Inference: The reproduction number R_t and infection period.

EX A2: SEIR model

Objective: Introduce an exposed compartment with a non-exponential residency time.

Implementation: 'comps' includes an exposed compartment with name "E", which has an Erlang distribution with shape parameter 2 (which is equivalent to two sequential compartments each with an exponential residency time of half the total).

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number R_t and latent and infectious periods.

EX A3: SIRD branching model

Objective: Introduce "branching", which describes a situation in which two (or more) transitions leave the same compartment.

Implementation: 'comps' includes an infected I compartment and 'trans' incorporates two transitions leaving this compartment.

Model: An SIRD branching model is used, in which infected individuals can either recover R or die D with associated branching probabilities. The reproduction number R_t is taken to be constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of S→I cases and I→R recoveries.

Inference: The reproduction number R_t, infectious period and branching probabilities.

EX A4: SEIAR branching model with infectious and asymptomatic states

Objective: Introduce branching and merging (which is when two compartments have transitions which merge back into a single compartment).

Implementation: 'trans' denotes two transition leaving the exposed E compartment (corresponding to two levels of infectivity) and two transitions merging back to a final recovered R state.

Model: The SEIAR model contains an exposed E compartment which branches such that infected individuals can either become symptomatic I or asymptomatic A with associated branching probabilities and relative infectivity. Individuals finally pass to a recovered R state. The reproduction number R_t is taken to be constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and the population in the R compartment.

Inference: The reproduction number R_t, the latent and infectious periods and symptomatic branching probabilities.

EX A5: SEIR model with initial conditions

Objective: Define initial state rather than assuming all individuals are susceptible.

Implementation: This example has no external force of infection. Rather the epidemic is initiated by specifying infected individuals in the initial state of the system. The initial state is specified in the "init_pop.csv" file in the data directory, and this is incorporated into the TOML file through 'init_pop'.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection. The population is demographically stratified by sex.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations, stratified by sex.

Inference: The reproduction number R_t and sex-stratified latent and infectious periods.

B) Data types and observation models

EX B1: SEIR model with different types of data

Objective: This provides an example which makes use of the four different types of data available in BEEP: transition data (here giving the weekly number of cases), population data (total population of infected individuals, measured weekly), marginal data (the total number of individuals which recover during a specified time window) and population fraction data (giving weekly estimates for the fraction of the population which are recovered).

Implementation: 'data_table' contains the four data source specifications with types "transition", "population", "marginal" and "population_fraction".

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection. The population is demographically stratified by sex.

Data: Weekly number of E→I cases, and measurements on the total infected I and population fraction of recovered individuals R, stratified by sex. Also the total recoveries I→R within a specified time window.

Inference: The reproduction number R_t and sex-stratified latent and infectious periods.

EX B2: SEIR model using fixed normal observation model

Objective: All the previous examples have used the "normal 10%" observation model, which assumes a normally distributed observation model with the standard deviation chosen to be a percentage of the measured value. Other possibilities exist within BEEP. This example looks at the case when a fixed standard deviation is applied to all observations from a given data source. The observation model can be observed in the visualisation software (visBEEP.html) by clicking on the "Data" tab and checking the "show observation model" box (note, observation models are not explicitly shown for ABC approaches, although they are still important because they inform the relative contribution in error coming from different observations).

Implementation: In 'data_tables' the 'obsmodel' property is set to "normal #" where # is the specified standard deviation in the observation model.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number R_t and latent and infectious periods.

EX B3: SEIR model using normal observation model loaded from file

Objective: Rather than assume a fixed standard deviation in observations (as in EX B2) here the standard deviations are explicitly loaded from a file. Specifically, for each data column in the data file there is a second column, with " SD" affixed to the heading, which provides the standard deviation in that data. For example, in the file "Data_EX_B3/S-I.csv" the column "Data" gives the weekly number of cases and "Data SD" gives the expected standard deviation in this quantity for the observation model.

Implementation: In 'data_tables' the 'obsmodel' property is set to "normal load".

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number R_t and latent and infectious periods.

EX B4: SEIR model using Poisson observation model

Objective: To apply a Poisson observation model, which assumes that the observed data values are Poisson distributed with mean given by the underlying value Y_i. This model implies that the variance is equal to the mean, and so, compared to the normal model characterised by a percentage (as used in most of the examples), tends to give relatively smaller standard deviations for higher values, *i.e.* a "good fit" corresponds to getting the peaks of time series correct (and less consideration is given to the fit of other points). It should be noted, however, that caution is required when using the Poisson observation model simultaneously on different data sources. For example, if population numbers are large and case rates low, much more weight will be placed on getting a good fit with the former than the latter.

Implementation: In 'data_tables' the 'obsmodel' property is set to "poisson".

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number R_t and latent and infectious periods.

EX B5: SEIR with only a fraction of cases observed

Objective: This example looks at a situation in which not all cases are observed. Specifically only a fraction 0.5 are assumed to be observed (this is somewhat akin to a sensitivity of a test).

Implementation: In 'data_tables' the 'factor' property is set to "0.5".

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number R_t and latent and infectious periods.

EX B6: SEIR with the fraction of cases observed varying with time

Objective: This example looks at the situation in which the fraction of observed cases varies with time. Specifically this fraction is assume to follow a piecewise linear spline (parameterised at five breakpoints times). The profile of this spline is inferred from the data.

Implementation: In 'data_tables' the 'factor_spline' property is set to the name of the spline, as specified by the user (in this case "fraction"). This spline is then specified in 'obs_spline'.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Inference: The time-varying estimated fraction of observed cases, the reproduction number R_t and latent and infectious periods.

EX B7: SEIR model using dates instead of numerical times

Objective: This replicates EX B6, but here time is represented by dates instead of numerically (*e.g.* the number of days). Dates can be used for defining the observation period, in data files or for defining spline breakpoints.

Implementation: 'time_format' defines the time format used. In this case it is set to "year-month-day" (*e.g.* such that dates take the form "2020-01-01"), but other possibilities exist.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of $E{\rightarrow}I$ cases and measurements on the total infected I and recovered R populations.

Inference: The reproduction number R_t and latent and infectious periods and the estimated fraction of observed cases.

C) Spatial and demographic stratification

EX C1: SIRD spatial model with area effects

Objective: This introduces a simple model for spatial stratification.

Implementation: 'areas' defines a file which provides information about the populations is four geographical regions. 'geo_mixing_matrix' defines a file which gives information about a 4×4 matrix that captures the relative mixing of individuals within and between regions (note, this matrix only needs to be specified up to a constant factor). Finally, 'area_effect' accounts for region-specific modifications to the disease transmission rate.

Model: An SIRD branching spatial model with four geographical areas: Region A, Region B, Region C, and Region D. The infected individuals can either recover or die, with associated branching probabilities. The reproduction number R_t remains constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of $S{\rightarrow}I$ cases and $I{\rightarrow}D$ deaths, separately for each region.

Inference: The reproduction number R_t (with a region-specific modifying factors), infectious period and branching probabili**ti**es.

EX C2: SIRD spatial model with area effects and covariate effects

Objective: On top of the model in EX C1, this analysis incorporates covariate effects, which relate properties of areas to the local disease transmission rate (in this case using population density and temperature).

Implementation: 'area_covars' is used to define covariates. Covariates appear in Eq.(4) and are related to the disease transmission rate through an exponential link function. The 'func' property can be used to transform the raw data before treating it as a covariate. This can either be set to "linear" or "log". In the case of "log", the covariate becomes the power to which the quantities is raised (so when applied to the population density a value zero or one implies frequency or density dependent transmission, respectively).

Model: An SIRD branching spatial model with ten geographical areas (Region A-J). The infected individuals can either recover or die with associated branching probabilities. The reproduction number R_t remains constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of S→I cases and I→R measurements, separately for the different regions.

Inference: The covariate factors, the reproduction number R_t (with region-specific modifying factors), infectious period and branching probabili**ti**es.

EX C3: SIRD model with demographic stratification

Objective: This provides an example of demographic stratification.

Implementation: 'democats' is used to divide the population into males and females. Model properties can be dependent on this stratification, *e.g.* under simulation the mean infectious period is defined by "Sex: 6 | 4", which here means that for males it is 6 days, whereas for females 4 days (note the order follow that defined in 'democats').

Model: An SIRD branching spatial model with the population demographically stratified into males and females. The infected individuals can either recover or die, with associated branching probabilities. The reproduction number R_t remains constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of S→I cases and I→D deaths, separately for the different sexes.

Inference: The reproduction number R_t and sex-stratified infectious period and branching probabilities.

EX C4: SIRD model with age stratification

Objective: This incorporates age stratification.

Implementation: 'ages' is used to define different age groups, here taken to be 0-19, 20-39, 40-59, and 60+. A contact matrix is specified by 'age_mixing_matrix', which gives the rate of effective contact between individuals within and between different age groups (note, this matrix only needs to be specified up to a constant factor). Model properties can be dependent on this stratification, *e.g.* under simulation the mean infectious period is defined by "age: 6 | 4 | 5 | 7" which here gives values for the different age groups (as defined in 'ages').

Model: An SIRD branching spatial model with age stratification. The infected individuals can either recover or die with associated branching probabilities. The reproduction number R_t remains constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of S→I cases and I→R measurements separately for the different ages.

Inference: The reproduction number R_t and age-stratified infectious period and branching probabilities.

EX C5: SIRD spatial model with factor affecting relative mixing within/between areas

Objective: This example looks at incorporating a factor (m_t in Eq.(8)) which affects the relative mixing between and within areas (specifically, $m_t = 0$ corresponds to no mixing between areas and $m_t = 1$ corresponds to mixing according to that specified by 'geo_mixing_matrix'). In this example m_t is assumed to be constant and in EX E4 later we look at incorporating time variation.

Implementation: 'geo_mixing_modify' sets the value for m_t in Eq.(8). Spatial results are plotted on a map of Scotland using boundary data specified in 'area_plot' (this sort of boundary data can often be found for geographical and administrative boundaries).

Model: An SIRD branching model with the population stratified into 32 Scottish local authorities. A contact matrix is specified which gives the rate of effective contact between individuals within and between different groups (note, this matrix only needs to be specified up to a constant factor). In

this case the matrix is estimated from census commuter data. The infected individuals can either recover or die with associated branching probabilities. The reproduction number R_t remains constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of S→I cases and I→D deaths, separately for the different areas.

Inference: The constant area mixing factor m_t, the reproduction number R_t, branching probabilities and infectious period.

D) Time-varying covariates

EX D1: SIRD model with time-varying covariate effects

Objective: The disease transmission rate is assumed to be modified by a factor, which on the log-scale has a linear dependency on a time-varying covariate (in this case temperature).

Implementation: 'tv_covars' is used to define time-varying covariates.

Model: An SIRD branching model in which infected individuals can either recover or die, with associated bra**n**ching probabilities. The base reproduction number R_t is taken to be constant, but disease transmission is modified to vary with a covariate (through an exponential dependency with the covariate multiplied by a fixed effect). A constant external force of infection is assumed.

Data: Weekly number of S→I cases and I→D deaths.

Inference: The fixed effect for the temperature covariate, reproduction number R_t, infectious period and branching probabili**ti**es.

EX D2: SIRD spatial model with time-varying area covariate effect

Objective: The disease transmission rate in each area is assumed to be modified by a factor, which on the log-scale has a linear dependency on a time-varying covariate within that area (in this case temperature).

Implementation: 'area_tv_covars' is used to define the values for time-varying covariates for each area.

Model: An SIRD branching model with ten geographical areas (Region A-J). In this infected individuals can either recover or die, with associated bra**n**ching probabilities. The base reproduction number R_t is taken to be constant, but disease transmission is modified to vary with the covariate (through an exponential dependency with the covariate multiplied by a fixed effect). A constant external force of infection is assumed.

Data: Weekly number of S→I cases and I→D deaths.

Inference: The fixed effect for the temperature covariate, reproduction number R_t, infectious period and branching probabili**ti**es.

EX D3: SIRD spatial model with level effects

Objective: The transmission rate in each area is assumed to be modified by a factor which gives the effect of a certain level applied to that area (where the level may change over time and is specified by the user), *e.g.* to mimic the effect of different levels of lock-down restrictions.

Implementation: 'level_effect' is used to define time-varying level effects.

Model: An SIRD branching model in which infected individuals can either recover or die, with associated bra**n**ching probabilities. The base reproduction number R_t is taken to be constant but it is modified to vary by a factor for each level. A constant external force of infection is assumed.

Data: Weekly number of S→I cases and I→D deaths.

Inference: The level effects, reproduction number R_t, infectious period and branching probabilities.

E) Time variation in transmission

EX E1: SIR model with time variation in the reproduction number R_t

Objective: A piecewise linear spline (determined by a series of model parameters at specified time points called "breakpoints") is used to represent the time variation in the reproduction number R_t.

Implementation: 'R_spline' is used to define time variation in the reproduction number.

Model: An SIR model is assumed with a constant external force of infection.

Data: Weekly number of S→I cases and I→R recoveries.

Inference: The time variation in the reproduction number R_t and infectious period.

EX E2: SIRD branching model with age stratification and time-variation in the contact matrix

Objective: This example looks at the case in which there is time variation in the relative mixing of different age groups.

Implementation: For each age group, the contact matrix is multiplied along the corresponding row and column by a time-varying 'age contact factor' represented by a spline.

Model: An SIRD branching model with the population demographically stratified into four age groups: 0-19, 20-39, 40-59, and 60+. A contact matrix is specified which gives the rate of effective contact between individuals within and between different age groups (note, this matrix only needs to be specified up to a constant factor). The infected individuals can either recover or die, with associated branching probabilities. The reproduction number R_t is approximated by a step function (*i.e.* it has a constant value before and after a putative lockdown event). A constant external force of infection is assumed.

Data: Weekly number of S→I cases and I→D deaths, separately for the different age groups.

Inference: The time-varying age contact factors, time variation in the reproduction number R_t and infectious period.

EX E3: SIR model with time variation in reproduction number R_t

Objective: Time variation in the external force of infection is added to the analysis. Note, due to confounding with the reproduction number it is usually not possible to infer this from the data. However, incorporating it, if known, leads to a more accurate estimate for the time variation in the reproduction number.

Implementation: 'efoi_spline' is used to define time variation in the external force of infection.

Model: An SIR model is assumed with a constant external force of infection. A piecewise linear spline is used to represent time variation in the reproduction number R_t.

Data: Weekly number of S→I cases and I→R recoveries.

Inference: The time variation in the reproduction number R_t and infectious period.

EX E4: SIRD spatial model with time-varying factor affecting relative mixing within/between areas

Objective: This example looks at incorporating a factor (m_t in Eq.(8)) which affects the relative mixing between and within areas (specifically, $m_t = 0$ corresponds to no mixing between areas and $m_t = 1$ corresponds to mixing as specified by 'geo_mixing_matrix'). In this example m_t is approximated by a

temporal curve (specified by the user, *e.g.* making use of transport data) multiplied by a factor which is estimated.

Implementation: 'geo_mixing_modify' sets the spline giving m_t. Spatial results are plotted on a map of Scotland using boundary data specified in 'area_plot' (this sort of boundary data can often be found for geographical and administrative boundaries).

Model: An SIRD branching model with the population stratified into 32 Scottish local authorities. A contact matrix is specified which gives the rate of effective contact between individuals within and between different groups (note, this matrix only needs to be specified up to a constant factor). In this case the matrix is estimated from census commuter data. The infected individuals can either recover or die, with associated branching probabilities. The reproduction number R_t remains constant (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of S→I cases and I→D deaths, separately for the different areas.

Inference: The constant multiplying factor used to generate the area mixing factor m_t, the reproduction number R_t, branching probabilities and infectious period.

F) Prediction, scenario and counterfactual analysis

EX F1: Prediction
Objective: Assuming data up to some point in time, here we want to predict what is going to happen next.

Implementation: First inference is performed on the data, *e.g.*

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F1.toml" mode="map"
```

which generates posterior samples[40] (for both model parameters θ and state ξ) in the "Posterior/samples/" directory. Note, the time window for this inference is defined by 'start' and 'end' in the TOML file. Next using these posterior samples we simulate the model forward in time beyond 'end':

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F1.toml" mode="prediction"
```

Here 'prediction_end = "100"' sets the time to simulate until. Opening "visBEEP.html" and going to the "Prediction→Data Table" tab on the menu we see the inherent variation in the number of weekly cases. Here the usual posterior distribution for states is showed until the 'end' time. Here we see a vertical dashed line with "Prediction start" beyond which are the future predictions from the model.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.

Prediction: Simulates system dynamics beyond available dataset.

---

[40] By default 400 posterior samples are generated, but this can be explicitly specified using 'nsample_final'.

EX F2: Prediction with time-varying spline

Objective: As with EX F1, we assume data up to some point in time and want to predict what is going to happen next. This example looks at the case in which there is the additional complication of a time-varying spline.

Implementation: First inference is performed on the data, *e.g.*

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F2.toml" mode="map"
```

which generates posterior samples (for both model parameters θ and state ξ) in the "Posterior/samples/" directory. Note, the time window for this inference is defined by 'start' and 'end' in the TOML file. Next, using these posterior samples we simulate the model forward in time beyond 'end':

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F2.toml" mode="prediction"
```

Here 'prediction_end = "450"' sets the time to simulate until. Opening "visBEEP.html" and going to the "Prediction→Data Table" tab on the menu we see the inherent variation in the number of weekly cases. Here the usual posterior distribution for states is showed until 'end'. The vertical dashed line with "Prediction start" denotes the time beyond which future predictions from the model are made. The "Prediction→Transmission→R" tab shows the time-variation in the spline for the reproduction number. We see that prior to 'end' this curve has been inferred from the data. However, since no data as available between 'end' and 'prediction_end' BEEP simply assumes the spline remains unchanged during this period (this is also generally true for any other spline defined in the model). In other words, predictions are produced under the implicit assumption that rates of contact in the population remain the same in the future as now (note, however, that changes in the susceptible fraction of the population are accounted for, as shown in the "Prediction→Transmission→R^{eff}" tab).

Model: An SIR model with a time-varying reproduction number and a constant external force of infection.

Data: Weekly number of S→I cases and I→R recoveries.

Prediction: Simulates system dynamics beyond available dataset assuming splines remain constant after 'end'.

EX F3: Scenario analysis

Objective: "Scenario analyisis" is when we look at what affect a future system modification will have on future predictions. As an example here we predict the effect of increasing transmission rate at some point in the future (*e.g.* mimicking the effect of lifting lockdown restrictions). Other types of system modification are shown in §3.10.

Implementation: First inference is performed on the data, *e.g.*

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F3.toml" mode="map"
```

which generates posterior samples. Note, the time window for this inference is defined by 'start' and 'end' in the TOML file. Next using these posterior samples we simulate the model forward in time beyond 'end':

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F3.toml" mode="prediction"
```

Here 'prediction_end = "100"' sets the time to make the future prediction until. 'modification' is set such that 10 days after the currently available data the rate of infected individuals is increased by a factor of two. Opening "visBEEP.html" and going to the "Prediction→Data Table" tab on the menu we see the inherent variation in the number of weekly cases. Here the usual posterior distribution for states is showed until the 'end' time. Beyond the vertical dashed line with "Prediction start" are the future predictions from the model. The modification applied to the system is within the second vertical dashed line on day 80.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of $E \rightarrow I$ cases and measurements on the total infected I and recovered R populations.

Scenario anakysis: Predicts future system dynamics assuming a future modification to the transmission rate.

EX F4: Counterfactual analysis

Objective: "Counterfactual analysis" is when we look at what affect past system modification would have had on the outcome up until the present. Here we image a situation in which no disease intervention was implemented (such that disease spread through the population based on an initial reproduction number R_0 until herd immunity is reached). We then consider what would have happened differently if a lockdown had been implemented. Other types of system modification are shown in §3.10.

Implementation: First inference is performed on the data, *e.g.*

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F4.toml" mode="map"
```

which generates posterior samples. Note, the time window for this inference is defined by 'start' and 'end' in the TOML file. Next using these posterior samples we simulate the model forward in time from when the mondification is instigated:

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F4.toml" mode="prediction"
```

'modification' is set such that on day 40 days the rate of infected individuals is reduced by a factor 0.3 (mimicking the effect of a reduced rate of contacts between individuals in a lockdown). Opening "visBEEP.html" and going to the "Prediction→Data Table" tab on the menu we see the inherent variation in the number of weekly cases. Beyond the vertical dashed line with "Prediction start" we see the effect of the lockdown, which is to dramatically reduce the number of cases.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of $E \rightarrow I$ cases and measurements on the total infected I and recovered R populations.

Counterfactual: Predicts how rates of disease would have been reduced if lockdown had been implemented.

EX F5: Multiple simulation

Objective: Simulating from a model multiple times using the same set of parameters allows the user to see how inherently stochastic the system is (in general stochasticity will tend to be less for larger population sizes, although it can still be important when the number of infected individuals is small early on in an epidemic).

Implementation: Multiple simulations can be carried out using the "multisim" mode. For example:

```
build/bin/BEEP inputfile="examples/EX_F5.toml" mode="multisim" nsimulation="400"
```

generates 400 state samples using a specified set of model parameters. Opening "visBEEP.html" and going to the "Multisim→Data Table" tab on the menu we see the inherent variation in the number of weekly cases (here the red dashed lines bound 95% of simulated outcomes). We find the simulated dataset (denoted by the solid black lines) lies within these bounds.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of $E \rightarrow I$ cases and measurements on the total infected I and recovered R populations.

Multiple simulation: The variation is state over multiple simulations from the same parameter values.

EX F6: Posterior predictive check

Objective: Posterior predictive checks are a way to check if simulating from the fitted model produces results similar to the observed data. They are helpful in assessing if a model gives valid predictions about reality - do they fit the observed data or not? (Note, they do not give a definitive answer on whether one model is better at fitting the data than another, however, they can help to check if a model makes sense.)

Implementation: First inference is performed on the data, *e.g.*

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F6.toml" mode="map"
```

which generates posterior samples (for both model parameters θ and state ξ) in the "Posterior/samples/" directory. Next the model can be simulated using the model parameter samples θ (disregarding the states ξ) using

```
mpirun -n 20 build/bin/BEEP inputfile="examples/EX_F6.toml" mode="prediction"
```

Here in the TOML file 'prediction_start = "0"' is set in to tell BEEP to start simulating at the beginning of the analysis period. Opening "visBEEP.html" and going to the "Prediction→State Output" tab on the menu we see the inherent variation in the number of weekly cases (here the red dashed lines bound 95% of simulated outcomes). Importantly we find the dataset (denoted by the solid black lines) lies within these bounds which tells us that the behaviour of the model is consistent with the real data. If this is not the case it implies that the model is not correctly capturing the true system dynamics. Note, the state bounds are bigger than in EX F4 because here we are taking into account uncertainty in parameter values, in addition to inherent stochasticity in the system.

Model: An SEIR model with constant reproduction number R_t (*i.e.* the contact rate between individuals remains the same over time), as well as a constant external force of infection.

Data: Weekly number of E→I cases and measurements on the total infected I and recovered R populations.
PPC: The variation is states over multiple simulations using posterior parameter samples.

G) Applications

EX G1 Model of Covid-19
Objective: Illustrate a model applicable to the sorts of publically available data sources for the 2020 Covid-19 pandemic.
Model: A realist compartmental model is needed to capture Covid-19. It contains the following compartments – S: Susceptible, E: Exposed, A: Asymptomatic, I: Infectious, T: PCR test-sensitive but non-infectious, C: Infected but non-infectious (due to self-isolation), H: Hospitalised, R: Recovered, and D: Dead. A piecewise linear spline is used to represent time variation in the reproduction number R_t.
Data: Weekly number of E→I cases, C→H hospital admissions, H→D deaths, PCR survey data (gives estimates for total infected population, *i.e.* the sum of those in A, I, C, T), and antibody survey data (gives estimates for total recovered population R). Note, this is simulated data using model parameters near to those found for the real data.
Inference: The time variation in the reproduction number R_t. Comparing for true curve with the inferred one we find excellent agreement. The only exception is at the beginning and end where the number of infected individuals is small (because of the lack of information with which to estimate reproduction number, the smoothing prior tends to smooth out fluctuations R_t). The time period in the test sensitive state (other time periods are set) and all branching probabilities.

EX G2: Age-structured Covid-19 Model
Objective: Introduce an age-structured model applicable to the sorts of publically available data sources for the 2020 Covid-19 pandemic.
Model: The population is stratified into 18 different age groups: 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44,45-49, 50-54, 55-59, 60-64, 65-69, 70-74, 75-79, 80+ and a group for care home residents. A realist compartmental model is needed to capture Covid-19. It contains the following compartments – S: Susceptible, E: Exposed, A: Asymptomatic, I: Infectious, T: PCR test-sensitive but non-infectious, C: Infected but non-infectious (due to self-isolation), H: Hospitalised (or potentially care home in the case of care home residents), R: Recovered, and D: Dead. A piecewise linear spline is used to represent time variation in the reproduction number R_t. The age contact matrix is derived from the BBC Pandemic survey.
Data: Age stratified weekly number of E→I cases, C→H hospital admissions, H→D deaths, PCR survey data (gives estimates for total infected population, *i.e.* the sum of those in A, I, C, T), and antibody survey data (gives estimates for total recovered population R). Note, this is simulated data using model parameters near to those found for the real data.
Inference: The time variation in the reproduction number R_t. Comparing for true curve with the inferred one we find excellent agreement. The only exception is at the beginning and end where the number of infected individuals is small (because of the lack of information with which to estimate

reproduction number, the smoothing prior tends to smooth out fluctuations R_t). The age-structured time period in the test sensitive state (other time periods are set) and all branching probabilities.

License and warranty

BEEP is free software under the terms of the GNU General Public License version 3 www.gnu.org/licenses/gpl-3.0.en.html. This allows users to redistribute and/or modify BEEP. The program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

Citing BEEP

We kindly request that those who do use BEEP analysis in their publications to cite this tool:

Pooley CM, Doeschl-Wilson AB, Marion G., *BEEP – A flexible tool for simulation, inference and prediction using compartmental models*. To be submitted (2021).

Acknowledgments

We would like to acknowledge the contribution of two libraries used within the software: toml11 (github.com/ToruNiina/toml11), which is a C++ TOML parser that BEEP uses to read in the input TOML file, and nlohmann/json (github.com/nlohmann/json), which is a JSON parser for converting geographic areas into visual outputs.

Appendix A: The reproduction number R_t

The model in Eq.(1) is parameterised in terms of a time-varying quantity R_t referred to as the "reproduction number". This appendix explains what this quantity is and how it is related to other model parameters. For clarity we ignore the effect of strain (although this could easily be incorporated below by simply changing every instance of R_t with $R_t\psi_s$[41]).

Definitions

We begin with a clarification of terminology:

Generation – An epidemic starts with a single infected individual. We refer to this as "generation 1". That individual then goes on to infect other secondary infections which make up generation 2. Those individuals subsequently infect generation 3, and so on and so forth.

Basic reproduction number R_0 – This is often defined to be the expected number of cases directly caused by the individual in generation 1 (*i.e.* in contact with an otherwise completely susceptible population). However, for models that include different demographic classifications, *e.g.* age and/or sex, or complex compartmental structures, careful consideration needs to be given to precisely how R_0 is calculated. In particular, just averaging over the demographic possibilities for the initially infected individual is not enough. One must also consider what happens in subsequent generations in the early phase of an epidemic to get a meaningful estimate for R_0 (because it typically takes

---

[41] ψ_s is the factor increase in reproduction number for a given strain s.

several generations for the distribution in demographic groups that cause the bulk of disease transmission to manifest). See below for how R_0 is actually calculated.

Time-varying reproduction number R_t – This sets the value R_0 would have taken if the initial rate of mixing between individuals in the population is taken to be the same as that at time t. As such, R_t should be interpreted as a quantity proportional to the rate at which individuals come into contact with each other (with each contact allowing for the possibility of disease transmission). So, for example, when R_t goes down it indicates that either individuals are meeting less frequently, or disease control are blocking transmission somehow, such as mask wearing. Note, disease transmission only actually occurs when contacts occur between infected and susceptible individuals. It is important to remember, therefore, that R_t *does not* account for the reduction in the susceptible fraction of the population as the epidemic progresses.

Time-varying effective reproduction number R_t^{eff} – The effective reproduction number is the expected number of cases directly caused by an infected individual as a function of time t. Note, this *does* take into account the fact that as the epidemic progresses the fraction of susceptible individuals reduces (causing effective transmission upon contact of individuals to become less and less common). R_t^{eff} is always less than R_t and if it reduces below 1 then herd immunity in reached (*i.e.* the disease will naturally die out over time). R_t^{eff} is an output from BEEP.

Relationship between reproduction number and transmission rate

Perhaps a more standard way to write the force of infection in Eq.(1) is in terms of a transmission rate parameter:

$$\lambda_{a,d,t} = \beta_t \sigma_d \alpha_{a,t} \sum_{a',d',c} M_{a,a',t} A_{d,d',t} i^c N_{a',d',t}^c, \tag{A30}$$

where here we ignore the external force of infection (as it doesn't play a role in the calculation of reproduction number). β_t is called the "transmission rate", and it is a proportionality constant that relates quantities measuring the general mixing of individuals in the population to the actual probability per unit time of an individual becoming infected (so β_t incorporates effects such as mask wearing, social distancing etc…). Comparing Eqs.(1) and (A30) we see that

$$\beta_t = r_t R_t. \tag{A31}$$

This equation simply states that the transmission rate is proportional to the reproduction number through a factor r_t. BEEP works by first parameterising a spline that represent R_t and then calculating r_t to obtain the force of infection in Eq.(1) (see below for how this is done in practice).

Calculating R_t

We outline here the approach taken by Diekmann *et* al. to calculate the reproduction number R_t (remembering the definition from above, that R_t is the value that R_0 would have taken assuming a disease transmission rate at time t).

First, a set of compartmental states Ω for which individuals are infected (but not necessarily infectious) is identified. For the purposes of explanation, we refer to the example SEIR

compartmental model in Fig. 1[42]. Here $\Omega = \{E_d, I_d\}$, where d goes over all demographic possibilities. We consider the case of two demographic groups (denoted M and F for male and female), but the results below can easily be extended for arbitrary d.

A vector $\mathbf{v} = (E_M, E_F, I_M, I_F)^T$ is defined[43] to be the number of individuals in each of the infected compartments in Ω. In the deterministic case the time evolution in $\mathbf{v}$ is given by

$$\frac{d\mathbf{v}}{dt} = (\mathbf{F}_t - \mathbf{\Sigma})\mathbf{v}, \tag{A32}$$

where $\mathbf{F}_t$ is a matrix accounting for the rate of individuals *entering* compartments contained in Ω, and $\mathbf{\Sigma}$ is a matrix accounting for transitions *between* and *leaving* compartments within Ω.

From Fig. 1 we see that individuals enter Ω through the exposed E_M and E_F states, caused by infectious individuals in the I state (this is derived from the force of infection in Eq.(A30)):

$$\mathbf{F}_t = \beta_t \left\langle \alpha_{a,t} \right\rangle_a \begin{bmatrix} 0 & 0 & \phi_M \sigma_M A_{M,M,t} i^I & \phi_M \sigma_M A_{M,F,t} i^I \\ 0 & 0 & \phi_F \sigma_F A_{F,M,t} i^I & \phi_F \sigma_F A_{F,F,t} i^I \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{A33}$$

where $\langle \ldots \rangle_a$ denotes an average of the area effects[44], ϕ_d gives the fraction of the population in demographic group d, σ_d gives the relative susceptibility, $\mathbf{A}$ is a (potentially time-varying) matrix giving the contact rate between demographic groups and, finally, i^c is the infectivity of compartment c.

The matrix $\mathbf{\Sigma}$, representing transitions between and leaving the infected states Ω, is given by:

$$\mathbf{\Sigma} = \begin{bmatrix} \frac{1}{m_M} & 0 & 0 & 0 \\ 0 & \frac{1}{m_F} & 0 & 0 \\ -\frac{1}{m_M} & 0 & \frac{1}{r_M} & 0 \\ 0 & -\frac{1}{m_F} & 0 & \frac{1}{r_F} \end{bmatrix}, \tag{A34}$$

where m_d and r_d are the mean residency times in the E and I states. This was constructed by considering each transition in the model in turn. Denoting the initial and final infected compartments to be i and j, matrix element Σ_{ii} gets a positive contribution given by the individual rate (because individuals are leaving state i) and Σ_{ji} gets a corresponding negative contribution (because those individuals are entering state j). Note, if j is not one of the infected states in Ω, this second contribution is ignored.

The inverse of the matrix in Eq.(A34) is given by

---

[42] For simplicity we take the case when the Erlang distribution is a simply exponential by setting $k=1$.
[43] The superscript "T" strands from transpose, and this converts a row vector into a column vector.
[44] This average is weighed by the populations in each of the areas a.

$$\mathbf{\Sigma}^{-1} = \begin{bmatrix} m_M & 0 & 0 & 0 \\ 0 & m_F & 0 & 0 \\ r_M & 0 & r_M & 0 \\ 0 & r_F & 0 & r_F \end{bmatrix}. \tag{A35}$$

This has a simple interpretation: If an individual enters state i, Σ_{ji}^{-1} gives the time, on average, that individual is expected to (eventually) spend in state j. For example, inspecting the first column in Eq.(A35) we see that if an individual enters state E_M (at the top) it will stay in E_M for time m_M, spend no time in E_F (individuals do not change their demographic state), time r_M in state I_M, and no time in state I_F.

We now construct the next generation matrix $\mathbf{K}_t$. We denote $\mathbf{w}_g$ to be a vector giving the number of individuals entering the different states in Ω in generation g. The equivalent vector for the next generation is given by[45]

$$\mathbf{w}_{g+1} = \mathbf{K}_t \mathbf{w}_g \quad \text{where} \quad \mathbf{K}_t = \mathbf{F}_t \mathbf{\Sigma}^{-1}. \tag{A36}$$

The reasoning behind this expression is as follows: Suppose we consider an individual in generation g that enters state i. As described above, during its infected lifetime this individual spends on average Σ_{ji}^{-1} in each state j. But in doing so Eq.(A33) tells us that through its own infectiousness it will generate F_{kj} new infected individuals in each of the states k. This means that $w_{g+1,k}$ gains a contribution $F_{kj}\Sigma_{ji}^{-1}$ for each of the $w_{g,i}$ individuals in generation g that enter state i. Summing over all the possible values for i and j gives the relationship in Eq.(A36).

Next we ask the question what happens when we iterate Eq.(A36) over many generations? It turns out that this equation can be solved[46], as shown in Appendix B. In summary, as the generations increase, so $\mathbf{w}_g$ becomes proportional to the normalised eigenvector $\mathbf{e}$ of the matrix $\mathbf{K}_t$ which has the largest eigenvalue. The value of this eigenvalue provides an estimate for R_t (by definition this is the average number of infections an individual in one generation causes in the next).

The eigenvector $\mathbf{e}$ itself has an intuitive explanation. Supposing that the mixing matrix $\mathbf{A}$ implies that certain demographic groups come into contact more often, *e.g.* younger age groups. Irrespective of the age of the person who initiates an epidemic, eventually those driving disease progression will be the young and so $\mathbf{e}$ will have proportionately larger values associated with entering the exposed state for young individuals.

The careful reader may wonder why the spatial elements in Eq.(A33) have been simply averaged over. An alternative would be to consider the system as a whole with each of the compartments in each of the spatial locations having its own states in Ω. This is indeed possible to do, however the stumbling block comes when one considers Eq.(A36). The reason lies in the fact that the number of

---

[45] In fact because individuals usually only enter Ω through a limited number of states (E_M and E_F in our example), then, without loss of generality, $\mathbf{w}$ and $\mathbf{K}$ can be defined for just these states. This is a commonly used technique to increase computational efficiency.
[46] We assume the timescale of the initial phase of the epidemic is slow compared to time variation in other model parameters, *e.g.* β_t.

generations for $\mathbf{w}$ to actually converge on the eigenvector $\mathbf{e}$ with largest eigenvalue becomes infeasibly large. It would be of the order of time taken for a single person to infect the rest of the country, and this cannot be reconciled with the fact that R_t should represent early epidemic behaviour. Consequently such an approach would seem ill-advised, yielding unrealistically high values[47] for R_t.

Calculating $r_{s,t}$

The previous section showed that given a transmission rate β_t an estimate for R_t could be made. BEEP, however, works the opposite way around. Parameters are used to inform a spline for R_t, and this in turn is used to calculate β_t. We now provide an description of how this is done.

First, we define the quantity

$$\mathbf{F}_t' = \frac{\mathbf{F}_t}{\beta_t}. \tag{A37}$$

From Eq.(A33) we see that $\mathbf{F}_t'$ is independent of β_t and so can be calculated using known model parameters. Next, we define a modified next generation matrix:

$$\mathbf{K}_t' = \mathbf{F}_t'\mathbf{\Sigma}^{-1}. \tag{A38}$$

The largest eigenvalue of this matrix gives R_t divided by β_t. From Eq.(A31) we see that taking the reciprocal of this quantity gives r_t. For systems with multiple strains, we calculate r_t separately for each strain s, and this defines $r_{s,t}$ in Eq.(1).

The reason BEEP parameterises R_t (rather than β_t) is twofold: Firstly it allows for realistic priors to easily be placed on the model (previous studies may have established that R_t is within some realistic range, *e.g.* 2-4), which is not true of β, and secondly it was found to be an effective way to improve MCMC mixing[48].

Calculating R_t^{eff}

As mentioned above, the effective reproduction number is the expected number of cases directly caused by an infected individual as a function of time t. This has to account for the fact that some susceptible individuals have already been infected, recovered and acquired immunity.

Equation (A33) is modified to now give

---

[47] In essence it would imply that R_t for the whole country would be almost the same as the area a with the highest R_t.

[48] Effectively it is a re-parameterisation of the model for which variables are not so highly correlated in the posterior.

$$\mathbf{F}'_{sus,t} = \frac{\mathbf{F}_{sus,t}}{\beta_t} = \langle \alpha_{a,t} \rangle_a \begin{bmatrix} 0 & 0 & \phi^{sus}_{M,t}\sigma_M A_{M,M,t} i^I & \phi^{sus}_{M,t}\sigma_M A_{M,F,t} i^I \\ 0 & 0 & \phi^{sus}_{F,t}\sigma_F A_{F,M,t} i^I & \phi^{sus}_{F,t}\sigma_F A_{F,F,t} i^I \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{A39}$$

where ϕ^{sus}_d is the fraction of the population in demographic group d that are *susceptible*. This can be used to calculate a new modified NGM:

$$\mathbf{K}'_{sus,t} = \mathbf{F}'_{sus,t}\mathbf{\Sigma}^{-1}. \tag{A40}$$

If we define the largest eigenvalue of $\mathbf{K}'_t$ in Eq.(A38) to be λ_t, and for $\mathbf{K}'_{sus,t}$ in Eq.(A40) to be $\lambda_{sus,t}$, the effective reproduction number can be related to R_t through

$$R_t^{eff} = \frac{\lambda_{sus,t}}{\lambda_t} R_t. \tag{A41}$$

A graph showing R_t^{eff} is automatically generated by BEEP.

Calculating the generation time

Another derived output from BEEP is the generation time. This is defined to be the time interval between when an individual becomes infected and the average time that person transmits their infection to others. This quantity is important because it specifies how fast infections are spreading within the community with the passing of each generation.

We now show how to calculate the generation time using the example from above. Based on the analysis in Appendix B, the fraction of individuals entering state i is given by the corresponding element in the eigenvector e_i. Equation (A35) shows a matrix Σ^{-1}_{ji} giving the average time those individuals then go on to spend in state j. Whilst in state j they initiate new infections in each of the compartments k at a rate given by F_{kj} (see Eq.(A33)). The average timing of those new infections is given by the elements of the following vector:

$$\boldsymbol{\tau} = \left(\tfrac{1}{2}m_M, \tfrac{1}{2}m_F, m_M + \tfrac{1}{2}r_M, m_F + \tfrac{1}{2}r_A \right)^T. \tag{A42}$$

Combining these contributions yields the final result

$$T_{gen} = \frac{\sum_{k,j,i} F_{kj}\tau_j \Sigma^{-1}_{ji} e_i}{\sum_{k,j,i} F_{kj}\Sigma^{-1}_{ji} e_i}. \tag{A43}$$

Appendix B: Solving iterative matrix equations

This appendix outlines the solution to the matrix equation in Eq.(A36). Note, we drop the index t because we assume that the initial spread of the disease is fast compared to the timescale over which model parameter changes (such as the transmission rate β_t).

We refer to **e** as an "eigenvector" and λ as an "eigenvalue" of a matrix **K** if it solves the equation

$$\boldsymbol{Ke} = \lambda\boldsymbol{e}. \tag{B1}$$

In general a square matrix of size N will actually have N eigenvectors $\mathbf{e}_j$ and eigenvalues λ_j, ordered such that λ_1 is the highest and λ_N is the lowest. These can collectively be written as

$$\mathbf{KU} = \mathbf{U}\begin{bmatrix} \lambda_1 & 0 & 0 & \cdots \\ 0 & \lambda_2 & 0 & \cdots \\ 0 & 0 & \lambda_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{B2}$$

where $\mathbf{U}=[\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3,\dots]$ is a matrix made up of the eigenvectors. Multiplying both sides of Eq.(B2) by the inverse matrix $\mathbf{U}^{-1}$ and substituting this expression for $\mathbf{K}$ into Eq.(A36) gives

$$\mathbf{w}_g = \mathbf{K}^{g-1}\mathbf{w}_1$$

$$= \left(\mathbf{U}\begin{bmatrix} \lambda_1 & 0 & 0 & \cdots \\ 0 & \lambda_2 & 0 & \cdots \\ 0 & 0 & \lambda_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}^{-1} \mathbf{w}_1 \right) \tag{B3}$$

$$= \mathbf{U}\begin{bmatrix} \lambda_1^{g-1} & 0 & 0 & \cdots \\ 0 & \lambda_2^{g-1} & 0 & \cdots \\ 0 & 0 & \lambda_3^{g-1} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

A key feature of this relationship is that as the generation number g increases, so the diagonal element that corresponds to the largest eigenvalue λ_1 dominates over all the others (which also means that $\mathbf{w}_g$ become proportional to the eigenvector $\mathbf{e}_1$). Because the overall number of individuals in $\mathbf{w}_g$ goes up by a proportion λ_1 each generation, so λ_1 provides an approximation to R_t.

Appendix C: Derivation of the geographical mixing matrix Z

This appendix explains how the geographical mixing matrix $Z_{a,a'}$ in Eq.(8), and incorporated into the model using the 'geo_mixing_matrix' TOML command (see §3.6.3), can be derived from data giving the movement of individuals within and between areas.

As an example we consider the census data from 2011. This splits up the geography of the UK into middle layer super output areas[49] (MSOAs) in England and Wales and intermediate zones[50] (IZs) in Scotland. To avoid confusion we collectively refer to these as IZs. The census flow data itself provides a matrix giving the number of individuals commuting from any given IZ to any other IZ. Analysis using BEEP, however, is usually performed on a coarser geographical scale (*e.g.* on a local authority level), so this must be accounted for.

---

[49] These are 7,201 areas covering England and Wales.
[50] These are 1,279 areas covering Scotland.
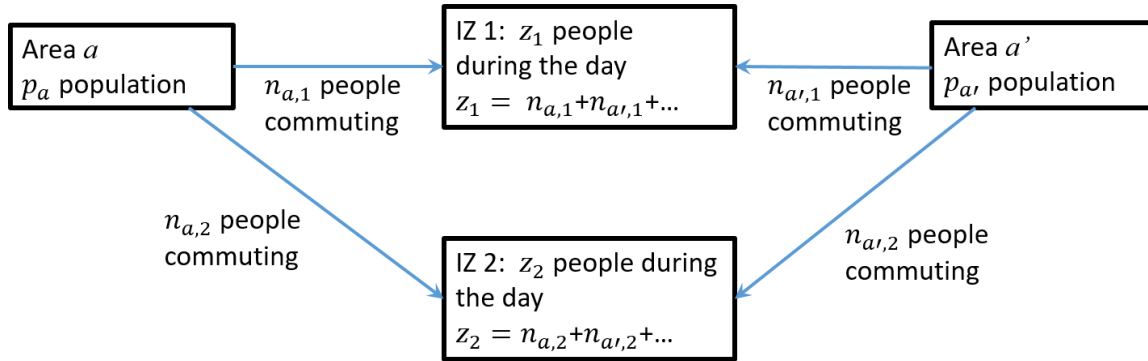
The diagram above illustrates the basic procedure. We consider two (potentially) different areas a and a' (*e.g.* local authorities) and consider the rate at which individuals come into contact. Each area a has a population p_a that can be estimated from census data. During the day people move to work in a way defined by the census flow data, where $n_{a,k}$ gives the number of individuals moving from area a to IZ k. Note, those people who don't commute are assumed to remain in their own IZ (by adding an appropriate contribution to $n_{a,k}$).

Suppose someone in area a is infectious. The probability of them travelling to IZ k is given by $n_{a,k}/p_a$. Assuming random mixing of individuals in IZ k, the probability of them passing on their infection to someone in area a' is proportional to $n_{a',k}/z_k$ (where z_k is the total number of people commuting to IZ k, see diagram). Consequently the force of infection from area a to a' via k is proportional to $(n_{a,k}/p_a)(n_{a',k}/z_k)/p_{a'}$. Summing this over all intermediate IZs k gives the final result:

$$Z_{a,a'} \propto \frac{1}{p_a p_{a'}} \sum_k \frac{n_{a,k} n_{a',k}}{z_k}. \tag{C1}$$

The matrix $\mathbf{Z}$ is normalised such that each infected individual is expected to infect the same number of secondary cases on average (geographical variation in force of infection explicitly enters the model through the area effects in Eq.(1)).

Note, the matrix in Eq.(C1) is symmetric, as would be expected since the overall number of contacts between individuals in different areas must be the same.

Appendix D: Derivation of age contact matrix

The matrix $C_{z,z'}$ used in Eq.(9) gives the rate of contacts between different age groups z in the population. Previous reports (*e.g.* Prem *et al.* [5], POLYMOD [3] and BBC Pandemic [4]) have published estimates for a different matrix $K_{z,z'}$, which gives the average number of contacts in age category z made by an individual in age category z' (note, following convention z denotes the row of the matrix and z' is the column). $\mathbf{K}$ is a matrix read into BEEP using the 'age_mixing_matrix' TOML command (see §3.5.1). We explain here how $\mathbf{C}$ is calculate from $\mathbf{K}$.

Suppose the number of people in each age category is given by N_z. Because $\boldsymbol{K}$ acts to generate a force of infection, so the raw numbers in $K_{z,z'}$ must be divided by N_z to ensure that infections are generated in proportion to the number of contacts:

$$C_{z,z'} = \frac{K_{z,z'}}{N_z}. \tag{D1}$$

Note, the total number of contacts between individuals in categories z and z' must be the same, *i.e.*

$$K_{z,z'}N_{z'} = K_{z',z}N_i. \tag{D2}$$

This means that whilst $K_{z,z'}$ is asymmetric, the matrix $\boldsymbol{C}$ is symmetric. This can be seen as follows:

$$C_{z,z'} = \frac{K_{z,z'}}{N_z} = \frac{C_{z',z}N_z}{N_z N_{z'}} = \frac{K_{z',z}}{N_{z'}} = C_{z',z} \tag{D3}$$

In reality, the matrix $\boldsymbol{C}$ generated above becomes asymmetric due to stochastic noise introduced by sampling used to generate the raw data in the first place. To reduce this noise we here enforce symmetry be updating $\boldsymbol{C}$ according to

$$C_{z,z'}^{\text{new}} = \tfrac{1}{2}(C_{z,z'} + C_{z',z}). \tag{D4}$$

Appendix E: Accounting for thresholds in the observation model

This appendix shows expressions for the observation model (see §2.3.4) when the data y_i is set to be thresholded, *i.e.* its precise value is unknown, but it is at or below some specified threshold value V (note, the value for V is set using the 'threshold' property within 'data_tables', as described in §3.7).

- **Normal** – If Y_i is less than or equal to the threshold value V the observation model has a constant maximal value (which makes sense because the observation and data are entirely consistent), whereas if Y_i is larger than V the observation model probability correspondingly goes down:

$$\left. \begin{array}{l} \pi\left(\text{Thresholded} \mid Y_i(\xi)\right) = N(V \mid V, \sigma_i) \\ \pi\left(\text{Thresholded} \mid Y_i(\xi)\right) = N(V \mid Y_i(\xi), \sigma_i) \end{array} \right\} \quad \begin{array}{l} \text{If } Y_i(\xi) \leq V \\ \text{If } Y_i(\xi) > V \end{array} \tag{E1}$$

where $N(x|\mu,\sigma)$ is normal probability for x given a mean μ and standard deviation σ. In cases in which the standard deviation σ_i is defined by a percentage p_i then the relationship is given by:

$$\sigma_i = 0.01 \times p_i \left(V(1-\varepsilon) + y_{\max}\varepsilon \right), \tag{E2}$$

where ε is a small constant (set to 0.05 by default).

- **Poisson** – In this case the observation model is calculated by simply adding up the probabilities for each possible value v up to V:

$$\pi\left(\text{Thresholded} \mid Y_i(\xi)\right) = \sum_{v=0}^{V} P(v \mid Y_i(\xi)), \tag{E3}$$

where $P(x|\mu)$ is the Poisson probability distribution for x given a mean μ.

- **Negative binomial** – Similarly:

$$\pi\left(\text{Thresholded} \mid Y_i(\xi)\right) = \sum_{v=0}^{V} NB(v \mid Y_i(\xi), k), \tag{E4}$$

where $NB(x|\mu,r)$ is the negative binomial probability distribution for x given a mean μ and shape parameter r.

Appendix F: Approximate likelihood calculation

This appendix describes how the likelihood is estimated.

outlines the methodology used for making a direct calculation of the likelihood. This is achieved by assuming that the potential stochastic paths the system can take are well approximated by a multivariate normal approximation in the compartmental populations. In particular the mean of this MVN is given by μ_c^t (at time t in compartment c) and the covariance is given by K_{cd}^t (between compartments c and d).

We now explain how the dynamic equations for $\boldsymbol{\mu}$ and $\mathbf{K}$ are derived. The starting point is the tau-leaping algorithm:

$$p_c^{t+\Delta t} = p_c^t + \sum_j T_{cj} n_j^t (\lambda_j^t), \qquad (5.5)$$

which shows how the population in p_c^t is changed at a later time point $t+\Delta t$. The sum j goes over all transitions within the system. n_j^t is the Poisson distributed number of transitions between t and $t+\Delta t$ with the mean transition number being given by λ_j^t. The matrix T_{cj} is a matrix that captures the effect transitions have on populations. This matrix typically contains element which are either -1 (if the transition j makes an individual leave compartment c), 1 (if it makes an individual enter compartment c), otherwise zero. For example, for the SIR model

$$T = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \qquad (5.6)$$

where the three rows represent and S, I and R and the two columns represent infection and recovery.

The mean transition numbers λ_j^t are themselves functionally dependent on the population p_c^t. This dependency can by Taylor expanded

$$\lambda_j^t(\mathbf{p}^t) \simeq \qquad \frac{d\lambda_j^t}{dp_d}\bigg|_{\mu^t} \qquad \dots \qquad (5.7)$$

The higher order terms are neglected as small (in reality there are quadratic terms in only the infection process because this depends on both the number of susceptible and infected individuals). Substituting this into Eq.(5.5) gives

$$p_c^{t+\Delta t} = p_c^t + \sum_j T_{cj} n_j^t \left(\lambda_j^t(\mathbf{\mu}^t) + \sum_d \frac{d\lambda_j^t}{dp_d}\bigg|_{\mathbf{\mu}^t} (p_d^t - \mu_d^t) \right). \qquad (5.8)$$

Taking the stochastic average of this expression (essentially replacing the Poisson distributed transition number n_j^t with its mean) gives us

$$\begin{aligned} \mu_c^{t+\Delta t} &= \left\langle p_c^{t+\Delta t} \right\rangle \\ &= \mu_c^t + \sum_j T_{cj} \lambda_j^t(\mathbf{\mu}^t). \end{aligned} \qquad (5.9)$$

The mean transition number λ_j^t can alternatively be represented by the transition rate r_j^t multiplied by the timestep Δt. Consequently Eq.(5.9) can be re-expressed (for the limit of small Δt) as

$$\frac{d\mu_c^t}{dt} = \sum_j T_{cj} r_j^t(\mathbf{\mu}^t), \qquad (5.10)$$

which is exactly the same expression that would be obtained assuming deterministic dynamics (*i.e.* ignoring stochasticity).

Subtracting Eq.(5.9) from Eq.(5.8)

$$p_c^{t+\Delta t} - \mu_c^{t+\Delta t} = p_c^t - \mu_c^t + \sum_j T_{cj}\left[n_j^t\left(\lambda_j^t(\boldsymbol{\mu}^t) + \sum_m \frac{d\lambda_j^t}{dp_m}\bigg|_{\boldsymbol{\mu}^t}(p_m^t - \mu_m^t)\right) - \lambda_j^t(\boldsymbol{\mu}^t)\right], \qquad (5.11)$$

Multiplying this for two different compartment c and d and taking the expectation (over all stochastic realisations) yields the evolution in the covariance matrix:

$$
\begin{aligned}
K_{cd}^{t+\Delta t} &= \left\langle (p_c^{t+\Delta t} - \mu_c^{t+\Delta t})(p_d^{t+\Delta t} - \mu_d^{t+\Delta t}) \right\rangle \\
&= \left\langle \left(p_c^t - \mu_c^t + \sum_j T_{cj}\left[n_j^t\left(\lambda_j^t(\boldsymbol{\mu}^t) + \sum_m \frac{d\lambda_j^t}{dp_m}\bigg|_{\boldsymbol{\mu}^t}(p_m^t - \mu_m^t)\right) - \lambda_j^t(\boldsymbol{\mu}^t)\right]\right) \right. \\
&\qquad \left. \left(p_d^t - \mu_d^t + \sum_k T_{dk}\left[n_k^t\left(\lambda_k^t(\boldsymbol{\mu}^t) + \sum_n \frac{d\lambda_k^t}{dp_n}\bigg|_{\boldsymbol{\mu}^t}(p_n^t - \mu_n^t)\right) - \lambda_k^t(\boldsymbol{\mu}^t)\right]\right) \right\rangle
\end{aligned}
\qquad (5.12)
$$

Remembering that the Poisson distributed transition numbers n_j^t are strictly independent with variance given by their mean, this expression can be simplified to (ignoring a term quadratic terms in the gradient of mean transition number λ_j^t, which is again assumed to be small)

$$K_{cd}^{t+\Delta t} = K_{cd}^t + \sum_j T_{cj}T_{dj}\lambda_j^t(\boldsymbol{\mu}^t) + \left(\sum_{j,m} T_{cj}\frac{d\lambda_j^t}{dp_m}\bigg|_{\boldsymbol{\mu}^t}K_{dm}^t\right) + \left(\sum_{k,n} T_{dk}\frac{d\lambda_k^t}{dp_n}\bigg|_{\boldsymbol{\mu}^t}K_{cn}^t\right). \qquad (5.13)$$

Again taking the small time-step limit and rearranging indices finally gives:

$$\frac{dK_{cd}^t}{dt} = \sum_j T_{cj}T_{dj}\lambda_j^t(\boldsymbol{\mu}^t) + \sum_{j,m}\left(T_{cj}K_{dm}^t + T_{dj}K_{cm}^t\right)\frac{d\lambda_j^t}{dp_m}\bigg|_{\boldsymbol{\mu}^t}. \qquad (5.14)$$

The observation model

This section looks at the model. Observations on the system restrict the possible states the posterior state can take. Two type of observation can be implemented using this methodology:

Population measurements – Supposing i indexes population observations and Y_i represents the population measured on compartments C_i (which may be one or a group of compartments) at time τ_i with uncertainty σ_i. We take the system state $\xi=\{\mathbf{p}^t\}$ to be defined by the dynamic variation in the populations. The observation probability for a single population measurement is given by

$$\pi(y_i \mid \xi) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{1}{2\sigma_i^2}\left(\left(\sum_{c\in C_i} p_c^{\tau_i}\right) - Y_i\right)^2}, \qquad (5.15)$$

which is a simple univariate normal distribution centred at the observation and with standard deviations σ_i.

Transition measurements – These are converted into effective population measurements on the population from which the transition originated. Supposing we know the number of transitions N_i within a given time interval τ (with standard deviation κ_i). If the interval is sufficiently small, this will provide a proxy for the population in the compartment from which the transition is originating. In particular if the per capita transition rate is r then the estimated population size will be

$$Y_i = \frac{N_i}{r\tau}, \qquad \sigma_i^2 = \frac{N_i + \kappa_i^2}{r^2\tau^2} \ . \tag{5.16}$$

Often several observations are made concurrently at the same time point (for example we might know both the number in hospital and the number dead at a series of time points). Therefore here we order observations into time "slices" indexed by s:

$$\pi(y\,|\,\xi) = \prod_s \pi(y_s\,|\,\xi) \tag{5.17}$$

and within each slice we take a product over all observations i within that slice:

$$\pi(y_s\,|\,\xi) = \prod_{i \in s} \pi(y_i\,|\,\xi). \tag{5.18}$$

Due to the fact that all the distributions in Eq.(5.15) are Gaussian, the overall probability distribution for a slice can be written as proportional to a multivariate normal distribution

$$\pi(y_s\,|\,\xi) = \omega_s f_{MVN}(\mathbf{p}^{t_s}\,|\,\mathbf{m}_s, \mathbf{\Omega}_s) \tag{5.19}$$

with mean $\mathbf{m}_s$, covariance matrix $\mathbf{\Omega}_s$ and constant of proportionality ω_s.

Calculating the likelihood

The likelihood is defined by the probability of the data y given a parameter set θ. Given some latent state ξ this is given by

$$L(y\,|\,\theta) = \int_\xi \pi(y\,|\,\xi)\pi(\xi\,|\,\theta)d\xi. \tag{5.20}$$

This section looks at how this likelihood can be estimated.

Usually the initial state for the system $\mathbf{\mu}(0)$ is known: it consists of all individuals in the susceptible compartment and all elements of the covariance matrix set to zero, *i.e.* $\mathbf{K}(0){=}\mathbf{0}$. Equations (5.10) and (5.14) can be integrated in time (in our case using a simple Euler integration scheme) up to the first observation time slice.

At this time we apply the observation model. The probability of being in population states immediately preceding the observation are given by taking the product of the estimated distribution in states and the observation

$$f_{MVN}(\mathbf{p}^{t_s}\,|\,\mathbf{\mu}^{t_s}, \mathbf{K}^{t_s}) \times \omega_s f_{MVN}(\mathbf{p}^{t_s}\,|\,\mathbf{m}_s, \mathbf{\Omega}_s). \tag{5.21}$$

The product of two multivariate normal distributions is also multivariate normal, and so this can be expressed as

$$L_s f_{MVN}\left(\mathbf{p}^{t_s} \mid \mathbf{m}_s, \mathbf{\Omega}_s\right),$$ (5.22)

where

$$\left(\mathbf{K}_{new}^{t_s}\right)^{-1} = \left(\mathbf{K}^{t_s}\right)^{-1} + \left(\mathbf{\Omega}_s\right)^{-1}$$

$$\left(\mathbf{K}_{new}^{t_s}\right)^{-1} \mathbf{\mu}_{new}^{t_s} = \left(\mathbf{K}^{t_s}\right)^{-1} \mathbf{\mu}^{t_s} + \left(\mathbf{\Omega}_s\right)^{-1} \mathbf{m}_s$$ (5.23)

define the mean $\mathbf{\mu}_{new}^{t_s}$ and covariance matrix $\mathbf{K}_{new}^{t_s}$ for the new distribution and the numerical coefficient is given by

$$L_s = \omega_s \sqrt{\frac{\left|\mathbf{K}_{new}^{t_s}\right|}{\left(2\pi\right)^d \left|\mathbf{K}^{t_s}\right|\left|\mathbf{\Omega}_s\right|}} e^{-\frac{1}{2}\left(\mathbf{\mu}^{t_s T}\left(\mathbf{K}^{t_s}\right)^{-1}\mathbf{\mu}^{t_s} + \mathbf{m}_s^T\left(\mathbf{\Omega}^{t_s}\right)^{-1}\mathbf{m}_s - \mathbf{\mu}_{new}^{t_s T}\left(\mathbf{K}_{new}^{t_s}\right)^{-1}\mathbf{\mu}_{new}^{t_s}\right)}.$$ (5.24)

This factor shows the fit between the states given by the model and the observed data at slice s.

The new values $\mathbf{\mu}_{new}^{t_s}$ and $\mathbf{K}_{new}^{t_s}$ replace the old ones and the system is then integrated up to the second time slice at which point the observations are accounted for. Sequentially repeating this procedure yields the final estimate

$$L(y \mid \theta) = \prod_s L_s.$$ (5.25)

Note, in practice due to finite numerical accuracy one usual deals in log likelihoods, and so this becomes a sum instead of a product.

NOTE: In practice the analysis above was modified in one crucial respect. It was noticed that the MVN distribution for the populations in Eq.(5.21) is, in fact, improper. The reason is that there are conservations which underlie the system dynamics. For example, for the SIR model S+I+R is fixed to the total number of individuals in the system. Because of this conservation it was necessary to first remove the susceptible S states from the matrices in Eq.(5.21) before the subsequent analysis was performed. The covariances and means of these S states were the reconstructed after each observation slice to ensure correct preservation of the conservations.

Appendix G: Calculation of the model evidence

This appendix explains how the model evidence from §2.3.6 is estimated for the different inference algorithms:

Metropolis coupled MCMC (MC³)

This algorithm runs K separate MCMC chains, indexed by k (where $k=1$ represents the posterior and $k=K$ represents the prior). The inverse temperature for each of these chains is given by

$$\phi_k = \left(\frac{K-k}{K-1}\right)^4 \phi_{post}.$$ (F1)

The model evidence is calculated using

$$\pi(y \mid \phi_1) = \prod_{k=2}^{K} \left(\frac{1}{N} \sum_{i=1}^{N} \pi(y \mid \xi_i^k)^{\phi_{k-1} - \phi_k} \right), \tag{F2}$$

where ξ_i^k are a series of N state samples (indexed by i) generated on chain k.

Particle annealed importance sampling using MBPs (PAIS-MBP)

This algorithm proceed in a series of generations indexed by g (where $g=1$ represent the prior and $g=G$ represents the posterior estimate). The model evidence is given by

$$\pi(y \mid \phi_G) = \prod_{g=1}^{G-1} \left(\frac{1}{N} \sum_{i=1}^{N} \pi(y \mid \xi_i^g)^{\phi_{g+1} - \phi_g} \right), \tag{F3}$$

where ξ_i^g are N state samples (indexed by i) generated in generation g (which has inverse temperature ϕ_g).

Approximate Bayesian Computation (ABC)

For a given error function cutoff EF_{cutoff}, the model evidence is simply given by the fraction of samples accepted.

Approximate Bayesian Computation sequential Monte Carlo (ABC-SMC)

This algorithm works in the following way. Suppose generation g contains N parameter samples θ_i^g indexed by i with weights w_i^g. A new parameter sample θ_{new} is generated from this by first selecting i in proportion to w_i^g and then sampling from a multivariate normal kernel distribution centred at θ_i^g with a covariance matrix approximation to the posterior Σ:

$$\theta_{new} \sim MVN(\theta_i^g, \Sigma). \tag{F4}$$

Considering this an application of importance sampling, then for θ_{new} to be considered a random sample from the prior $\pi(\theta)$ it must be weighted according to

$$w_{new} = \frac{\pi(\theta_{new})}{\sum_{i=1}^{N} w_i^g p_{MVN}(\theta_{new} \mid \theta_i^g, \Sigma)}, \tag{F5}$$

where p_{MNV} is the multivariate normal probability density function for the kernel.

A new state ξ_{new} is simulated from the model using θ_{new}. If the error function for ξ_{new} is less than EF_{cutoff}, θ_{new} becomes a new sample to be used in the next generation.

An estimate for the model evidence[51] is given by:

$$\pi(y \mid EF_{\text{cutoff}}) = \frac{\sum_{\text{new accepted}} w_{new}}{\sum_{\text{new}} w_{new}}, \tag{F6}$$

---

[51] This expression is, in fact, applicable to any generation, but is usually most accurate for the last.

where the top sum goes over the weights of only accepted parameter samples, and the bottom sums over all samples.

Approximate Bayesian Computation with model-based proposals (ABC-MBP)

This algorithm proceed in a series of generations indexed by g (where $g=1$ represent the prior and $g=G$ represents the posterior estimate). Because phase space is divided in two each time a new generation is added, the model evidence for a given generation g is approximately given by 2^g. However, a slightly more accurate expression, which uses all samples from all generations, can be calculated in the following way

$$\pi(y \mid EF_{\text{cutoff}}^{G}) = \prod_{g=1}^{G-1} \left[\frac{\sum_{g'=1}^{g} \sum_{i=1}^{N} H\left(EF_{\text{cutoff}}^{g'+1} - EF(\xi_i^{g'})\right)}{\sum_{g'=1}^{g} \sum_{i=1}^{N} H\left(EF_{\text{cutoff}}^{g'} - EF(\xi_i^{g'})\right)} \right], \tag{F7}$$

where ξ_i^{g} are N state samples (indexed by i) generated in generation g and H is the Heaviside step function (this gives a contribution of one if the error function on the sample is smaller than the cut-off).

Glossary: An index of all TOML commands

This provides an alphabetical list of all the commands that can be used in the input TOML file:

Command	Description
age_mixing_ matrix **§3.5.1**	Sets the name of a file containing an age-stratified contact matrix $\boldsymbol{K}$ (see Appendix I for further details). This square matrix must have the same dimensions as the number of age categories specified in 'ages' (if ages has only one category, 'age_mixing_matrix' does not need to be specified). Specifically, matrix element $K_{z,z'}$ gives the number of contacts in age category z made by an individual in age category z' (note, following convention z denotes the row of the matrix and z' is the column). *E.g.* age_mixing_matrix = "BBC-Pandemic.csv"

age_mixing_ modify
 	
§3.5.1
	This is used to modify the basic mixing matrix as a function of time (here the file "amm.csv" informs a spline, see §3.3).
	
	E.g. age_mixing_modify = [{type="row_column", agecat="age70+",
	    bp="[bp:amm.csv]", value="[value:amm.csv]", prior="Uniform(0,5)",
	    smooth_type="log_smooth", smooth="0.3"}]
	
	type – Determines the type of modification. Currently only one is supported:
	
	• "row_column" – This multiplies a row and column for a specified age category in the contact matrix $C_{z,z'}$ (see Appendix I) by a factor specified in a subsequently defined spline.

	agecat – Specifies the age category to which the modification is applied.
ages **§3.5.1**	Specifies the age categories used in the analysis. *E.g.* ages = {cats="age0-19 | age20-69 | age70+", sus_value="0.7 | 1.0 | *", sus_prior="MDir(0.5)"} *cats* – Sets the age categories (separated by '|'). *sus_X* – Defines the relative susceptibility for different age categories.
area_plot **§5.2**	Defines how areas are plotted using BEEP's visual tool. *E.g.* area_plot = { boundary="Scotland_areas.geojson", projection="equirectangular"} *boundary* – Provide the name a file which gives information about the boundaries of areas (currently '.geojson' and '.kml' geographical file formats are supported). *x_column / y_column* – Defines the columns in the 'areas' file which give the mean x and y position of the areas, *e.g.* mean longnitude and latitude (note this is an alternative to 'boundary' in cases in which in which this data is unavailable). *projection* – Sets the functional transformation of coordinates when plotting. Either "uniform" (by default) or "equirectangular" (suitable when displaying longitude/latitude data).
area_covars **§3.6.4**	Defines covariates for areas. These are used for estimating the contribution coming from fixed quantities within areas potentially related to disease transmission. *E.g.* area_covars= [{name="density", param="b1", value="0.1", prior="Uniform(0,1)", func="log"}] *name* – The name of the covariate (note this column heading must appear in the 'areas' file). *X* – Defines the parameter to be estimated. *func* - Sets the functional transformation to the raw data ("linear" or "log").
area_tv_covars **§3.6.4**	Defines time-varying covariates for areas. These are used for estimating the contribution coming from time-varying quantities within areas potentially related to disease transmission. *E.g.* area_tv_covars = [{name="Tarea", file="areatvcovar.csv",

	param="b2", value="0.1", prior="Uniform(0,1)", func="linear"}] *file* – Defines a file giving geographical covariate information as a function of time. *X* – Defines the parameter to be estimated. *func* - Sets the functional transformation to the raw data ("linear" or "log").
area_effect **§3.6.4**	These represent factors that modify the relative rate of transmission in different areas. *E.g.* area_effect = {value="[area effect:Scotland_LA.csv]", prior="MDir(0.5)"} *X* – Defines the parameters to be estimated. A Dirichlet prior is used to ensure the average area effect is 1 (see §3.2).
areas **§3.6.1**	Sets the filename of a table giving information about geographical areas (this can either be in tab-separated '.txt' text format or '.csv' format). *E.g.* areas= "areadata.csv" The table contains the following column: "area" (which gives a code or name that uniquely identifies each area), other coarser geographical scales if they are used (*e.g.* 'nhs region' amalgamates local authorities into NHS regions), columns for covariates in the model (*e.g.* population density)[52] and, potentially, the initial population make-up (see §3.6.2).
comps **§3.4.1**	Defines compartments in the model. *E.g.* comps = [{name="S"}, {name="E", dist="Erlang", k="2", mean_value="4"}, {name="I", dist="Exp", mean_value="4", inf_value="1"}, {name="R"}] *name* – Sets the name of a compartment. *mean_X* – Defines the model parameter(s) used for the mean waiting time (this can depend on demographic category, see §3.4.2). *inf_X* – Sets a model parameter that specifies the relative infectivity for that compartment (if not set it is assumed to be zero).
cutoff	Specifies the cut-off in the error function ('abc' mode).

---

[52] The columns do not have to have any particular order, provided they have the correct headings. Other columns may also exist which are not used by the analysis.

§4.2.4	*E.g.* cutoff = 1
cutoff_final **§4.2.2, §4.2.5**	Sets the final error function cut-off for the 'abc-mbp' mode (optional and by default set to zero corresponding to the prior). *E.g.* cutoff_final = 0.1
cutoff_frac **§4.2.4**	Specifies the fraction of accepted samples in the 'abc' mode. *E.g.* cutoff_frac = 0.01
datadir **§3.1**	Sets the directory where the data files are stored. *E.g.* datadir = "examples/Data_EX1"
data_tables **§3.7**	Provides information about the files that contain epidemiological data (multiple files are comma separated inside the square brackets). *E.g.* data_tables = [{file="EI.csv", type="transition", observation="E->I", timestep="7"}] *file* – The name of the file. This must be either a tab-separated '.txt' text file or a '.csv' file. *type* – Indicates the type of data: • "transition" – Time series information about the number of individuals moving down a transition (or transitions). • "population" – Time series population numbers within a specified compartment (or compartments). • "population_fraction" – Time series measurements for the population fraction within a specified compartment (or compartments). • "marginal" – The total number of transitions over a time period, typically stratified by a demographic classification. *observation* – Describes what is observed. When the type is "transition" or "marginal" this is given by the initial compartment followed by "->" and then the final compartment (additionally, multiple transitions can be specified, *e.g.* "I->D,I->R", would give the sum down both transitions). For "population" or "population_fraction" types, this is simply the name of the compartment under measurement (or several compartments that are comma separated). *obsmodel* – This sets the observation model which relates the observed data to the system state. Five possibilities exist: "normal #%" (where '#' is a number giving the percentage error), "normal #" (where '#' is an absolute

	standard deviation), "normal load" (where the standard deviation is loaded from a file), "poisson", and "negbin" (see §2.3.4 for details). In the case of "negbin" then "shape" must also be specified.
	geo_filt – This give the geography over which the observation are made (optional). The name for this corresponds to a specified column in the 'areas' file.
	geo_dep – This optional specification allows for multiple columns of data each referring to a given geographic group (*e.g.* 'geo_dep="area"' would give columns for each of the geographical areas).
	democats_filt – Filters to include only specified demographic groups (*e.g.* 'democats_filt="Sex:Male"' would mean that observations are made on only males).
	democats_dep – This optional specification allows for multiple columns of data each referring to a given demographic group (*e.g.* 'democats_dep="Sex"' would imply that the data has two columns, one for 'Male' and one for 'Female'). For age dependency 'democats_dep="age"' is used.
	start – Sets the first measurement date for the data (optional and by default taken from the data table).
	end – Sets the last measurement date for the data (optional and by default taken from the data table).
	shift – Allows for the raw data's date/time to be shifted by a specified number.
	timestep – An integer which sets the time step for observations.
	threshold – Sets the threshold value (see 'threshold_str').
	factor – Sets a factor which multiplies the value given in 'observation' to get to the data, *e.g.* this can be used to incorporate a test sensitivity (optional, by default set to 1).
	factor_spline – Uses a spline to provide the factor between the value given in 'observation' and the observed data.
	line_colour – Sets the colour of the line in the final plot. This can take the values "black", "red", "blue", "green", "yellow", "cyan", "magenta" (optional, by default set to red).
democats **§3.5.2**	Sets one or more demographic classifications. *E.g.* democats = [{name="Sex", cats="Male | Female"}, {name="Ethnicity", cats="White | Black | Asian"}]

	name – Sets the name for the demographic classification.
	cats – Sets the different categories within the classification.
	sus_X – Defines a parameter giving the relative susceptibility.
democat_change **§3.5.4**	This allows for the proportions within a specified demographic classification in the susceptible population to vary over time (for example, this could be used to study the effect of vaccination). *E.g.* democats = [{name="Vac Status", cats="Vac | Not Vac"}] democat_change = [{name="Vac Status", file="vac.csv"}] *name* – References the name for the demographic classification (set in 'democats'). *file* – The name of the file providing information about the change. This must be either a tab-separated '.txt' text file or a '.csv' file. *democats_filt* – Filters to include only specified demographic groups (*e.g.* setting this to "Sex:Male" would mean that the specified changes are made only on males). *geo_filt* – This give the geography over which the observation are made (*e.g.* setting this to "area:S12000005" would mean that the specified changes are made only on individuals in this particular area).
description **§3.9**	This can be used to add a description pf the analysis. This description is added to the "Model" tab. *E.g.* dynamics = "deterministic" Note, double spaces get converted to paragraph breaks.
dynamics **§3.9**	Used to determine the underlying system dynamics and can take the values "stochastic" or "deterministic" (optional, by default set to "stochastic"). *E.g.* dynamics = "deterministic" When run in deterministic mode, model inference methods that rely on MBPs cannot be used.
efoi_spline **§3.4.4**	Defines linear splines used to represent the external force of infection. *E.g.* efoi_spline = [{param="phi", value="1", bp="[bp:phi_spline.txt]", factor="[fac:phi_spline.txt]"}]

	See §3.3 for generally how splines are defined. Also: *geo_filt* – Specifies the area on which the spline acts (optional, set to all areas by default). *age_dist* – Specifies the fraction of individuals in different ages groups becoming infected as a result of the external force of infection (optional, by default they are set proportional to the population sizes in each demographic group). *strain* – If more than one strains exists in the model this specifies which one is being referred to (optional, set to all strains by default).
end **§3.1**	The ending time for the simulation or inference (in days or as a date). *E.g.* end = "2020-12-31"
geo_mixing_ matrix **§3.6.3**	This sets a files used to define the geographic mixing matrix between different regions. *E.g.* geo_mixing_matrix = "census_flow_data.csv"
geo_mixing_ modify **§3.6.3**	Defines a spline that informs the relative rate of mixing of individuals between regions. Specifically, this parameter multiplies the diagonal elements specified in the 'geo_mixing_matrix'. *E.g.* geo_mixing_modify = [{bp="[bp:gmm.csv]", value="[value:gmm.csv]", prior="Uniform(0,1)"}] See §3.3 for how splines are defined.
init_pop **§3.6.2**	Specifies a file giving information about the make-up of the initial population. *E.g.* init_pop = "initpop.csv",
invT **§4.2.6, §4.2.7**	Sets the inverse temperature for the posterior in the 'pmcmc' and 'mcmcmbp' modes. *E.g.* invT = 1.0
invT_final **§4.2.3, §4.2.8**	Sets the final inverse temperature (a) in the 'pais' mode or (b) for the highest temperature chain in the 'mc3' mode (optional and by default set to zero corresponding to the prior). *E.g.* invT_final = 0.1
invT_start **§4.2.8**	Sets the inverse temperature for the highest temperature chain in the 'mc3' mode (optional and by default set to zero corresponding to the prior). *E.g.* invT_start = 0.01

invT_power **§2.3.2**	Sets the power used to determine inverse temperatures in Eq.(16) (optional, by default set to 4). *E.g.* invT_power = 6
inputfile **§1.4**	Sets the input TOML file. *E.g.* inputfile="examples/EX1.toml"
level_effect **§3.6.3**	Allows different areas to have different levels of disease transmission (this can be used to incorporate levels of disease intervention, for example lockdown vs social distancing vs nothing). *E.g.* level_effect = {file="level.csv", param="level1 | level2", value="0.5 | *", prior="MDir(0.5)"} *file* – The name of a file providing information about the levels (this must be a table with rows giving different dates, columns giving different areas and the table elements themselves set to one of the values in 'param'). *X* – Sets parameters that represent the relative disease transmission rate for different levels.
mcmc_update	Sets which proposals are used for the MCMC updates *E.g.* mcmc_update = { full_mvn = "off", mvn = "on", single="on", dist_R_joint="on", demo_spec="on", mean_time="on", neighbour="on", joint="on", mvn_multiple="off", multiple_factor=1.0} Used only in tuning and testing the algorithm.
mode **§4.1**	This selects the mode of operation when BEEP is run. *E.g.* mode = "sim" The following possibilities exist: "sim" – Simulates from the model. "multisim" – Simulated multiple times to generate an ensemble of outputs. "prediction" – Performs model prediction (based on posterior samples). "abcmbp" – Performs inference using the ABC-MBP algorithm. "pais" – Performs inference using the PAIS-MBP algorithm. "abc" – Performs inference using the ABC algorithm. "abcsmc" – Performs inference using the ABC-SMC algorithm.

	"pmcmc" – Performs inference using the PMCMC algorithm.
	"mcmcmbp" – Performs inference using the MCMC-MBP algorithm.
	"mc3" – Performs inference using the MC^3 algorithm.
modification **§3.10**	In 'prediction' mode this specifies the change or changes made to the model. *E.g.* modification = [{start="2020-4-01", type="trans_rate_fac", trans="S->E", factor="0.5"}] *start* – Sets the start time for the change (optional, by default set to 'start' used in the analysis). *end* – Sets the end time for the change (optional, by default set to 'end' used in the analysis). *type* – Denotes the type of change. This can be one of the following types: • "trans_rate_fac" – This changes the rate of a specified transition by a factor ('trans' and 'factor' must be set). • "beta_fac" – Changes the transmission rate by a factor ('factor' must be set). • "efoi_fac" – Changes the external force of infection by a factor ('factor' must be set). • "spline_fac" – Multiplies a spline by a factor ('name' must correspond to 'name' set in a spline and 'factor' must be set). • "spline_set" – Sets the value of a spline ('name' must correspond to 'name' set in a spline and 'factor' must be set). *factor* – Specifies the factor for the modification (if needed). *value* – Specifies the value for the modification (if needed). *name* – Specifies the name of the spline affected (if needed). *trans* – Specifies the transition on which the change applies. *strain* – Specified if only a particular strain is referred to (optional). *geo_filt* – Specified to only applies to a particular geographical area (optional). *democats_filt* – Specified to only apply to a particular demographic category or set of categories (optional).
nburnin	Sets the number of 'burn-in' steps when performing MCMC (used in the 'pmcmc', 'mcmcmbp' and 'mc3' modes).

§4.2.6-8	*E.g.* nburnin = 100
	By default burn-in is set to a quarter of the number of samples 'nsample'
nchain **§4.2.8**	Sets the number of chains used when performing MC^3 in the mode 'mc3' *E.g.* nchain = 20
ngeneration **§4.2.2, §4.2.3,** **§4.2.5**	Sets the number of generations (required in the 'abcmbp', 'abcsmc' and 'pais' modes). *E.g.* ngeneration = 40
nodata_str **§3.7.2**	Defines the string used in files to represent the fact that no data is available (optional, by default set to ".") *E.g.* nodata_str = "NA"
nparticle **§4.2.2, §4.2.3,** **§4.2.6**	The number of particles (required in the 'abcmbp', 'pais' and 'pmcmc' modes) *E.g.* nparticle = 100
nquench **§4.2.8**	The number of iterations used in quenching the system when using the 'mcmcmbp' or 'mc3' modes (optional, set to a half of 'nburnin' by default). *E.g.* nquench = 100
nrun **§4**	Sets the number of runs which are performed in parallel (optional, used for inference and set to one by default). *E.g.* nrun = 4
nsample **§4.2.4-8**	Sets the number of samples to be generated (used in the 'abc', 'abcsmc', 'pmcmc', 'mcmcmbp' and 'mc3' modes). *E.g.* nsample = 200
nsimulation **§4.1.2**	Sets the number of simulations in 'multisim' mode. *E.g.* nsimulation = 100
nsim_per_sample **§4.3**	Sets the number of simulations per posterior sample in 'prediction' mode (optional, by default set to 4). *E.g.* nsim_per_sample = 10
nthin **§4.2.6-8**	Sets the thinning of samples in the 'pmcmc', 'mcmcmbp' and 'mc3' modes. This is used to save computational memory such that not every system sample is stored (which is typically not necessary because samples are highly

	correlated in these approaches). By default set to one thousandth of the number of samples. *E.g.* nthin = 10
nupdate **§4.2.2, §4.2.3**	Sets the number of MCMC 'updates' used per generation in the 'abcmbp' and pais' modes (by default set to 1). Here an update performs MBPs on each of the model parameters as well as combinations of them. *E.g.* nupdate = 2
obs_spline **§3.7.3**	Defines linear splines that are used as factors to relate the 'observation' quantity in 'data_tables' with the actual data observed. *E.g.* data_tables = [{type="transition", observation="E->I", file="IH.csv", factor_spline="obs1"}] obs_spline = [{name="obs1", value="0.1|0.3|0.5", prior="Uniform(0,1)", bp="start|20.03.20|end"}] This implies that the fraction of actual E->I transitions observed varies, starting at 10% and ending at 50%. As well as being able to set this spline, it can also be inferred from the data. See §3.3 for how splines are defined.
outputdir **§3.1**	Sets the name of the output directory (optional, set to "Output" by default). *E.g.* outputdir = "Output_EX1"
output_prop **§3.11.3**	Sets properties of the output. *E.g.* output_prop = {probdist="kde", h="5"} *probdist* – Sets how probability distributions are displayed. Either "kde" for kernel density estimation or "bin" for binning. *nbin* – Determines the number of bins used when binning (optional, set to 200 by default). *h* – Sets a smoothness parameter when using kernel density estimation (optional, set to 10 by default).
quench_factor **§4.2.3**	Sets the rate of quenching in the 'pais' mode (optional, set to 0.5 by default such that on average half of all particles each generation are discarded). *E.g.* quench_factor = 0.1
plot_param_values	Places simulated values onto output plots (optional, by default set to false).

§3.11.5	*E.g.* plot_param_values = "true"
prediction_end **§4.3**	Sets the end time (optional, by default set to inference end time). *E.g.* prediction_end = "2022-02-01"
prediction_start **§4.3**	In 'prediction' mode this sets the start time (optional, by default set to either when the first modification is made to the model or the inference end time, whichever is earlier). *E.g.* prediction_start = "2021-05-01"
prior_order **§3.2**	This command is used to specify any ordering of the parameters. *E.g.* R_spline = [{param="R0 | R1 | R2 | R2", value="2.0 | 1.9 | 1.5 | 0.6", prior="Uniform(0.4,4)"}] prior_order = "R0 > R1 | R1 > R2 | R2 > R3" In this example, the ordering ensures that the reproduction number is constrained to go down, consistent with the prior belief that implementing intervention strategies reduces disease transitions.
prop_size **§4.2.5**	Sets the proposal size in the 'abcsmc' mode (optional, set to 1 by default). This factor scales the MVN estimate of the posterior distribution in the previous generation to set the parameter proposal kernel. *E.g.* prop_size = 1.5
prob_reach **§3.11.2**	The probability of reaching a given compartment if infected. For example if set to the dead "D" compartment this can be used to calculate the infection fatality rate. *E.g.* prob_reach = [{name="ifr", comp="D"}] *name* – The name as generated in the output file. *comp* – The compartment reached.
region_effect	Set if there is a regional effect incorporated at a particular geographical scale. *E.g.* region_effect = {geography="area", sigma_value="0.3"} *geography* – The geography over which the regional effect acts (optional, set to all areas by default). *sigma_X* – The standard deviation in the regional effect.
R_spline **§3.4.3**	Defines the linear spline used to represent time variation in the reproduction number R_t.

	E.g. R_spline = [{bp="start | 2020-03-01 | 2020-07-15 | end", param="R0 | R1 | R2 | R3", value="2.0 | 1.5 | 0.5 | 1.0", prior="Uniform(0.4,4)", smooth_type="log_smooth", smooth = "0.5" }] *name* – Specifies the name of the spline (optional, by default set to "R_t"). *bp* – The times at which the breakpoints occur (note these must begin and end with the 'start' and 'end' times set for analysis. *X* – Sets parameters used for each spline point. *smooth_type* – Sets the type of smoothing used. The possible values are "no_smooth", "smooth" and "log_smooth" (optional, set to "no_smooth" by default). *smooth* – The strength of smoothing. *factor* – A multiplicative factor which multiplies the value of the parameter on each spline point (optional). *geo_filt* – Specifies the area(s) on which the spline acts (optional, set to all areas by default). See §3.3 for further details on defining splines.
seed **§4.1.1**	This is a positive integer used to set the initial value of the pseudorandom number generator. Note, because the model is stochastic in nature, changing this seed will lead to a different simulated results. Varying it can also be used to change the initial conditions for the inference algorithms (useful when checking for consistent convergence on the true posterior). *E.g.* seed = 10
start **§3.1**	Sets the start time for the simulation or inference (in days or as a date, as specified in 'time_format'). *E.g.* start = "2020-01-01"

state_outputs
 	
§3.11.1
	This allows for posterior graphs of the state to be plotted.
	
	E.g. state_outputs = [
	    {plot_name="Dynamics", type="population", observation="S"},
	    {plot_name="Dynamics", type="population", observation="E"},
	    {plot_name="Dynamics", type="population", observation="I"},
	    {plot_name="Dynamics", type="population", observation="R"}]
	
	This generates a single plot with title "Dynamics" that contains four curves representing the different compartmental populations in an SEIR model.

	type – Indicates the type of observation: • "transition" – Time series information about the number of individuals moving down a transition (or transitions). • "population" – Time series population numbers within a specified compartment (or compartments). • "population_fraction" – Time series measurements for the population fraction within a specified compartment (or compartments). • "marginal" – The total number of transitions over a time period, typically stratified by a demographic classification. *observation* – Describes what is observed. When the type is "transition" or "marginal" this is given by the initial compartment followed by "->" and then the final compartment (additionally, multiple transitions can be specified, *e.g.* "I->D,I->R", would give the sum down both transitions). For "population" or "population_fraction" types, this is simply the name of the compartment under measurement (or several compartments that are comma separated). *geo_filt* – This give the geography over which the observation are made (optional). The name for this corresponds to a specified column in the 'areas' file. *democats_filt* – Filters to include only specified demographic groups (*e.g.* 'democats_filt="Sex:Male"' would mean that observations are made on only males). *line_colour* – Sets the colour of the line in the final plot. This can take the values "black", "red", "blue", "green", "yellow", "cyan", "magenta" (optional, automatically generated by default). *plot_name* – Sets the name of the final plot (optional). Note, multiple lines can be put on the same plot by giving them the same '*plot_name*'. *label* – Uses to set the key label in the final plot (optional).
state_uncertainty **§3.11.6**	Sets how state uncertainty is displayed. To plot posterior means along with 95% credible intervals indicated by dashed lines (*e.g.*, see Fig. 6) then set this to "CI" (used by default). To plot individual posterior samples using faint blue lines (or alternatively plot multiple simulations in "mutisim" mode) then set to "curves". *E.g.* state_uncertainty = "curves"
steps_per_unit **_time**	Sets the number of discrete time steps performed per unit time (optional, set to 4 by default). Larger values mean the tau leaping algorithm better

| §3.9 | approximates the continuous time Gillespie algorithm, but take computationally longer to run. |
	E.g. steps_per_unit_time = 4
strains	Used to specify different strains of virus.
§3.8	*E.g.* strains = {cats="s1 | s2", sus_value="0.7 | *", sus_prior="MDir(0.5)", Rfactor_value="1.0 | 2.0", Rfactor_prior="Fixed(1) | Uniform(0.5,2)"}
	cats – Sets the names of the names of the strains.
	sus_X – Used to vary the susceptibility of individuals to different strains (optional).
	Rfactor_X – This gives the factor by which R_t increases for the different strains (usually one would be fixed to a value 1 to represent the reference strain).
threshold_str	When data contains thresholded values, this sets the string used in the data to represent this fact (optional, by default set to "*").
§3.7.2	*E.g.* threshold = "<=5"
	If now in 'data_tables' we set 'threshold="5"' then values of "<=5" in the data file are interpreted as at or below 5.
time_format	This determines the format in which times are represented in the input TOML file and any data files.
§3.1	*E.g.* time_format = "year-month-day"
	The following possibilities exist:
	• "*year-month-day*" – This uses the format "2020-03-01" to represent 1st March. • "day/month/year" – This uses the format "01/03/2020" to represent 1st March.
	number – An integer is used to represent the time (*e.g.* the number of days since some reference time point in the past).
time_labels	Sets time labels used to represent specific times of interest in the analysis. Not only can these be used subsequent instead of dates (*e.g.* when specifying the breakpoints in splines), but they are also placed as vertical reference lines on the final time-based output graphs.
§3.11.4	• *E.g.* time_labels = [{name="Lockdown", time="2020-03-23"}]

trans	Defines transitions between compartments.
§3.4.1	*E.g.* trans = [{from="S", to="E", infection="yes"}, 　　　　　{from="E", to="I"}, 　　　　　{from="I", to="R", prob_value="0.9", prob_prior="Dir(*)"}, 　　　　　{from="I", to="D", prob_value="*", prob_prior="Dir(*)"}] *from* – Sets the initial compartment. *to* – Sets the final compartment. *infection* – Specifies the infection transition (only one may be set). *prob_X* – Specifies the probability of moving down a given branch (note, this only needs to be specified when the 'from' compartment has more than one transition leaving it). Because the sum of branching probabilities for all transitions leaving a compartment must be one, so one of these probabilities does not need to be specified (because it can be calculated from the others). This is indicated by using the '*' symbol. Values can be set separately for different demographic groups (see §3.4.2). If not fixed, a Dirichlet prior is used during inference (see §3.2).
trans_combine	When using transition data, this combines data values until they are above this specified value (by default this functionality is turned off). This can help inference when data is highly stochastic and transition rates are low *E.g.* trans_combine =50
tv_covars **§3.6.4**	Defines time-varying covariates. These are used for estimating the contribution coming from time-varying quantities potentially related to disease transmission. *E.g.* tv_covars = [{name="covar", file="tvcovar.csv", param="b2", 　　　　　value="0.1", prior="Uniform(0,1)", func="linear"}] *name* – The name of the covariate (this heading must appear in the specified 'file'). *file* – Defines a file giving geographical covariate information as a function of time. *X* – Defines the parameter to be estimated. *func* - Sets the functional transformation to the raw data ("linear" or "log").

References

[1]   H. Heesterbeek *et al.*, "Modeling infectious disease dynamics in the complex landscape of global health," *Science,* vol. 347, no. 6227, 2015.

[2]   O. Diekmann, J. A. P. Heesterbeek, and J. A. Metz, "On the definition and the computation of the basic reproduction ratio R 0 in models for infectious diseases in heterogeneous populations," *Journal of mathematical biology,* vol. 28, no. 4, pp. 365-382, 1990.

[3]   J. Mossong *et al.*, "Social contacts and mixing patterns relevant to the spread of infectious diseases," *PLoS medicine,* vol. 5, no. 3, p. e74, 2008.

[4]   P. Klepac *et al.*, "Contacts in context: large-scale setting-specific social mixing matrices from the BBC Pandemic project," *MedRxiv,* 2020.

[5]   K. Prem, A. R. Cook, and M. Jit, "Projecting social contact matrices in 152 countries using contact surveys and demographic data," *PLoS computational biology,* vol. 13, no. 9, p. e1005697, 2017.