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 7

1.1 Compartmental epidemiological models 7

1.2 This software 8

1.3 Getting started 9

1.4 A ten-minute guide 10

1.4.1 Simulation example 10

1.4.2 Inference example 12

1.4.3 Prediction example 14

2 Models, data and inference 15

2.1 Epidemiological model 15

2.1.1 Stratification of the population 15

2.1.2 The force of infection 15

2.1.3 System dynamics 19

2.2 Data 20

2.3 Bayesian inference 20

2.3.1 Maximum *a posteriori* (MAP) estimation 21

2.3.2 Power posterior 21

2.3.3 Approximate Bayesian computation 22

2.3.4 Observation models 23

2.3.5 Incomplete observations 24

2.3.6 Model evidence 25

2.3.7 Which inference algorithm should I use? 26

3 Inputs into BEEP 27

3.1 A simple example 27

3.2 Priors 29

3.3 Splines 30

3.3.1 Simulation 30

3.3.2 Inference 31

3.3.3 Different types of spline 32

3.3.4 Using a file to define a spline 33

3.3.5 Other spline properties 33

3.4 Setting up a compartmental model 33
3.4.1 Compartments and transitions 33
3.4.2 Demographic dependency 34
3.4.3 Specifying the reproduction number R_t 34
3.4.4 Specifying the external force of infection 35
3.5 Defining demographic groups 35
3.5.1 Age stratification 35
3.5.2 Other demographic classifications 36
3.5.3 Variation in susceptibility 36
3.5.4 Incorporating changes in demographic groups 36
3.6 Spatial models 37
3.6.1 Loading area information 37
3.6.2 The initial population 38
3.6.3 Geographical mixing 38
3.6.4 Area effects 39
3.7 Data tables 41
3.7.1 Geographical and demographic filtering 41
3.7.2 Missing and thresholded data 42
3.7.3 Partial observation 42
3.8 Different disease strains 43
3.9 System dynamics 43
3.10 Model modification 43
3.11 Tailoring Outputs 44
3.11.1 Generating state outputs 44
3.11.2 Probability of reaching a specified compartment 44
3.11.3 Probability distribution plots 45
3.11.4 Setting time labels 45
3.11.5 Adding simulated values to graphs 45
3.11.6 Display state curves 45
4 Running BEEP 46
4.1 Simulation-based approaches 46
4.1.1 Single Simulation 46
4.1.2 Multiple simulation 46
4.2 Inference 46

4.2.1 Maximum *a posteriori* probability (MAP) using CMA-ES 46
4.2.2 Approximate Bayesian Computation with model-based proposals (ABC-MBP) 47
4.2.3 Particle annealed sampling with model-based proposals (PAS-MBP) 48
4.2.4 Approximate Bayesian computation (ABC) 48
4.2.5 Approximate Bayesian computation with sequential Monte Carlo (ABC-SMC) 48
4.2.6 Particle Markov chain Monte Carlo (PMCMC) 49
4.2.7 Markov chain Monte Carlo (MCMC-MBP) 49
4.2.8 Metropolis-coupled Markov chain Monte Carlo (MC^3) 49
4.3 Prediction 50
4.4 Parallelisation 51
5 Outputs from BEEP 51
5.1 Files generated 51
5.2 Graphical interface 52
5.2.1 Maps 52
6 Examples 53
EX: Exemplar used in §1.4 53
A) Introductory examples 53
EX A1: SIR model 53
EX A2: SEIR model 54
EX A3: SIRD branching model 54
EX A4: SEIAR branching model with infectious and asymptomatic states 54
EX A5: SEIR model with initial conditions 55
B) Data types and observation models 55
EX B1: SEIR model with different types of data 55
EX B2: SEIR model using fixed normal observation model 55
EX B3: SEIR model using normal observation model loaded from file 56
EX B4: SEIR model using Poisson observation model 56
EX B5: SEIR with only a fraction of cases observed 56
EX B6: SEIR with the fraction of cases observed varying with time 57
EX B7: SEIR model using dates instead of numerical times 57
C) Spatial and demographic stratification 57
EX C1: SIRD spatial model with area effects 57
EX C2: SIRD spatial model with area and covariate effects 58
EX C3: SIRD model with demographic stratification 58

EX C4: SIRD model with age stratification ... 58
EX C5: SIRD spatial model with factor affecting relative mixing within/between areas ... 59
EX C6: SIRD model with different disease strains ... 59
D) Time-varying covariates ... 59
EX D1: SIRD model with time-varying covariate effects ... 59
EX D2: SIRD spatial model with time-varying area covariate effect ... 60
EX D3: SIRD spatial model with level effects ... 60
E) Time variation in transmission ... 60
EX E1: SIR model with time variation in the reproduction number R_t ... 60
EX E2: SIRD branching model with age stratification and time-variation in the contact matrix ... 60
EX E3: SIR model with time variation in the external force of infection η_t ... 61
EX E4: SIRD spatial model with time-varying factor affecting relative mixing within/between areas ... 61
F) Prediction, scenario and counterfactual analysis ... 62
EX F1: Prediction ... 62
EX F2: Prediction with time-varying spline ... 62
EX F3: Scenario analysis ... 63
EX F4: Counterfactual analysis ... 64
EX F5: Multiple simulation ... 64
EX F6: Posterior predictive checking ... 65
G) Applications ... 65
EX G1 Model of Covid-19 ... 65
EX G2: Age-structured Covid-19 Model ... 66
EX G3: Age-structured Covid-19 Model with time variation in age contact ... 66
License and warranty ... 67
Citing BEEP ... 67
Acknowledgments ... 67
Appendix A: The reproduction number R_t ... 68
Definitions ... 68
Relationship between reproduction number and transmission rate ... 68
Calculating R_t ... 69
Calculating $r_{s,t}$ ... 71
Calculating $\boldsymbol{R_t^{\mathrm{eff}}}$ ... 72
Calculating the generation time ... 72

Appendix B: Solving iterative matrix equations 73
Appendix C: Derivation of the geographical mixing matrix Z 74
Appendix D: Derivation of age contact matrix 75
Appendix E: Accounting for thresholds in the observation model 76
Appendix F: Approximate likelihood calculation 76
Observations 79
Combining simultaneous observations 79
Calculating the likelihood 80
Appendix G: Calculation of the model evidence 81
Maximum *a posteriori* estimate (MAP) 81
Metropolis coupled MCMC (MC^3) 81
Particle annealed sampling using MBPs (PAS-MBP) 82
Approximate Bayesian Computation (ABC) 82
Approximate Bayesian Computation sequential Monte Carlo (ABC-SMC) 82
Approximate Bayesian Computation with model-based proposals (ABC-MBP) 83
Glossary: An index of all TOML commands 84
References 100

1 Introduction

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

The manual is structured in the following way: this introductory section describes compartmental models and sketches out the main features of BEEP, along with an easy-to-follow guide for performing a simple analysis. Section 2 describes, from a mathematical perspective, the types of models and data that can be incorporated into BEEP, and provides some details on how inference is performed. Section 3 focuses on the practicalities of how these models and data are defined as inputs into BEEP. Section 4 looks at the various options available when running BEEP, and, finally, section 5 describes the outputs BEEP generates. Some 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]).

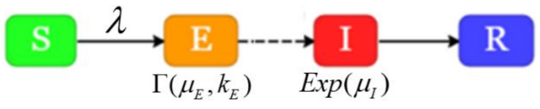


Figure 1. Example compartmental model. This describes how an individual's infection status changes over time. S: susceptible, E: exposed, I: infectious, R: recovered. Individuals are infected at a rate given by the force of infection λ, undergo a non-infectious gamma distributed incubation period (with mean time μ_E and shape parameter k_E), remain infectious for an exponentially distributed period (with mean μ_I), and finally recover.

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 potential infections coming from outside the system under study). 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 fluctuation 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 all these various possibilities is discussed in sections 2 and 3 later.

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

To clarify terminology used in the manual:

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).

Transition – Represents the movement of an individual between two compartments (denoted by the arrows in Fig. 1).

Model parameters θ – Quantities in the model that determine how individuals behave (*e.g.* in Fig. 1 θ would include μ_E, k_E and μ_I, as well as other parameters incorporated into λ).

State ξ – Represents system dynamics over a time period under consideration. This is composed of the initial compartments as well as any transition times for all the individuals in the population.

1.2 This software

BEEP incorporates three modes of operation:

Simulation – Given a set of model parameters θ, potential realisations of the state ξ 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 θ are estimated from available data (along with associated uncertainties in these estimates). BEEP accepts a variety of different data 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 estimate 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 (counterfactual analysis).

Once run, BEEP displays outputs via an 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, scenario and counterfactual analysis 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:

- Efficient parallel code (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 "Message Passing Interface" (MPI) installed to compile and run this code. This can be loaded using the command:

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

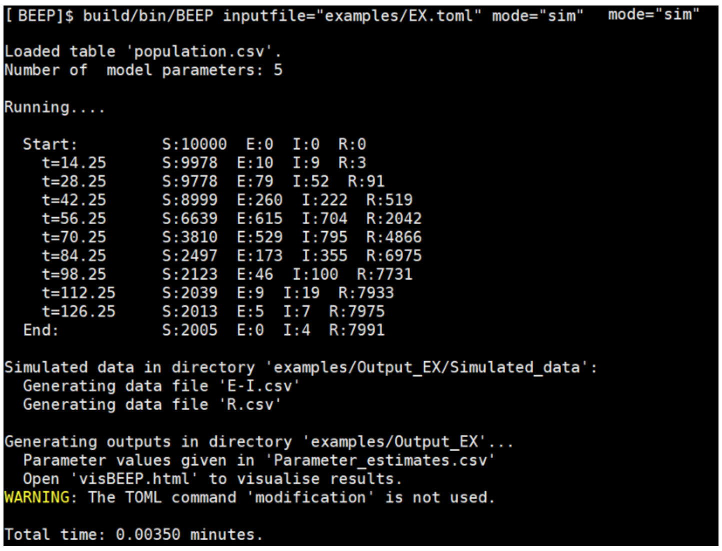
STEP 3: Build – The project is built using cmake (minimum version 3.13 required). Navigate to the "BEEP" directory and use the command:

```
cmake -Bbuild
```

STEP 4: Compile – The code is compiled using:

```
cmake --build
```

BEEP is now ready to use.



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

Loaded table 'population.csv'.
Number of  model parameters: 5

Running....

  Start:          S:10000  E:0  I:0  R:0
    t=14.25       S:9978  E:10  I:9  R:3
    t=28.25       S:9778  E:79  I:52  R:91
    t=42.25       S:8999  E:260  I:222  R:519
    t=56.25       S:6639  E:615  I:704  R:2042
    t=70.25       S:3810  E:529  I:795  R:4866
    t=84.25       S:2497  E:173  I:355  R:6975
    t=98.25       S:2123  E:46  I:100  R:7731
    t=112.25      S:2039  E:9  I:19  R:7933
    t=126.25      S:2013  E:5  I:7  R:7975
  End:            S:2005  E:0  I:4  R:7991

Simulated data in directory 'examples/Output_EX/Simulated_data':
  Generating data file 'E-I.csv'
  Generating data file 'R.csv'

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

Total time: 0.00350 minutes.
```

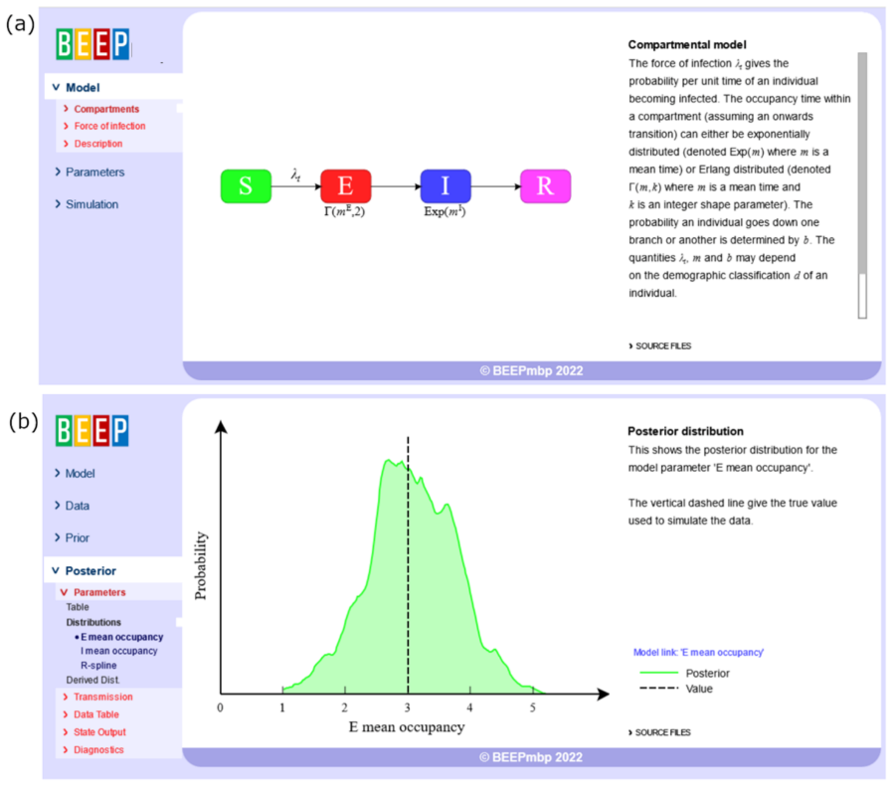
Figure 2. Terminal output for a simulation. This shows the terminal output from BEEP when running the example in §1.4.1. First, the file "population.csv" is loaded (which provides information about the population under study, in this case a single area containing 100,000 individuals). Next the actual 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)).

1.4 A ten-minute guide

Here we provide a ten-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 the SEIR model in Fig. 1 and specifies 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 output files 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 (*e.g.* see Fig. 4).
- In the sub-directory "Simulated_data" the files "E-I.csv" and "R.csv" have been 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 within a one week interval) and the total number of recovered individuals, again measured weekly.



Compartmental model

The force of infection λ_t gives the probability per unit time of an individual becoming infected. The occupancy time within a compartment (assuming an onwards transition) can either be exponentially distributed (denoted $\mathrm{Exp}(m)$ where m is a mean time) or Erlang distributed (denoted $\Gamma(m,k)$ where m is a mean time and k is an integer shape parameter). The probability an individual goes down one branch or another is determined by b. The quantities λ_t, m and b may depend on the demographic classification d of an individual.

› SOURCE FILES

Posterior distribution

This shows the posterior distribution for the model parameter 'E mean occupancy'.

The vertical dashed line give the true value used to simulate the data.

› SOURCE FILES

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 from the simulation (see Fig. 4 for examples).

(b) Inference – Further tabs on the menu can be selected to provide details about the data, prior and posterior (example plots are shown in Fig. 7).

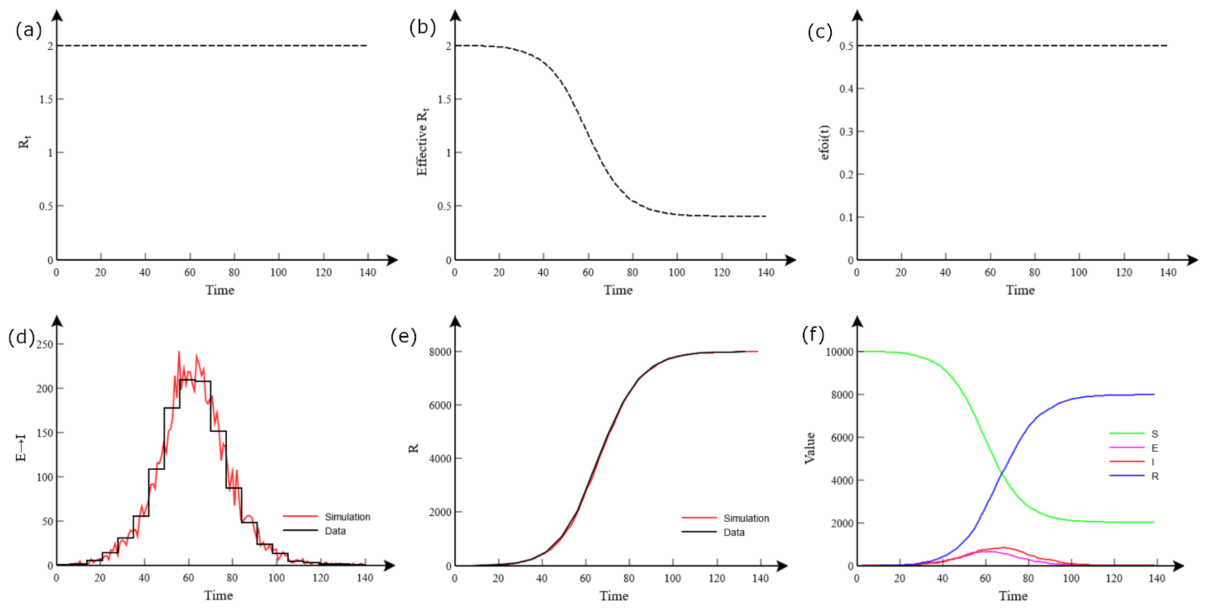
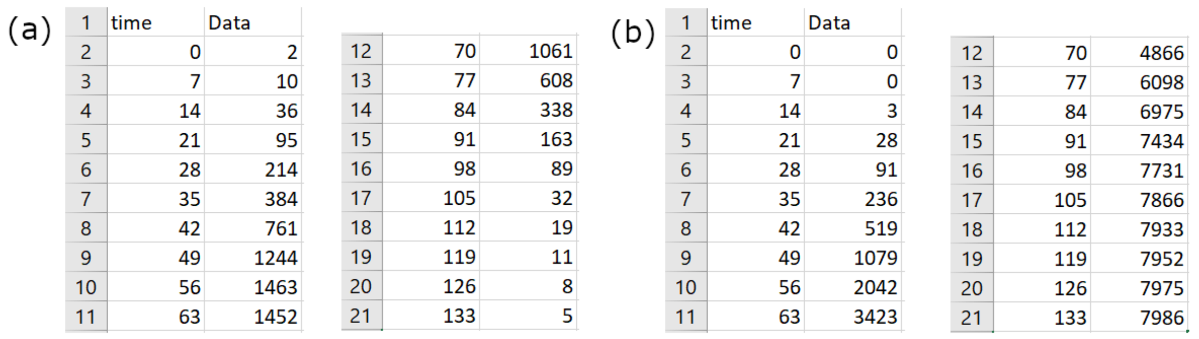


Figure 4. Simulation from model. This shows outputs from BEEP when simulating from the SEIR model in Fig. 1. (a) The reproduction number over time (which here has a constant value of 2). (b) The effective reproduction number (which starts at 2, but drops down as the epidemic progresses because of depletion of the susceptible population). (c) The external force of infection (which is here taken to have a constant value of 0.5 meaning that on average 0.5 individuals per unit time become infected as a result of contacts made with those 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.



(a)

1	time	Data
2	0	2
3	7	10
4	14	36
5	21	95
6	28	214
7	35	384
8	42	761
9	49	1244
10	56	1463
11	63	1452
12	70	1061
13	77	608
14	84	338
15	91	163
16	98	89
17	105	32
18	112	19
19	119	11
20	126	8
21	133	5

(b)

1	time	Data
2	0	0
3	7	0
4	14	3
5	21	28
6	28	91
7	35	236
8	42	519
9	49	1079
10	56	2042
11	63	3423
12	70	4866
13	77	6098
14	84	6975
15	91	7434
16	98	7731
17	105	7866
18	112	7933
19	119	7952
20	126	7975
21	133	7986

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.

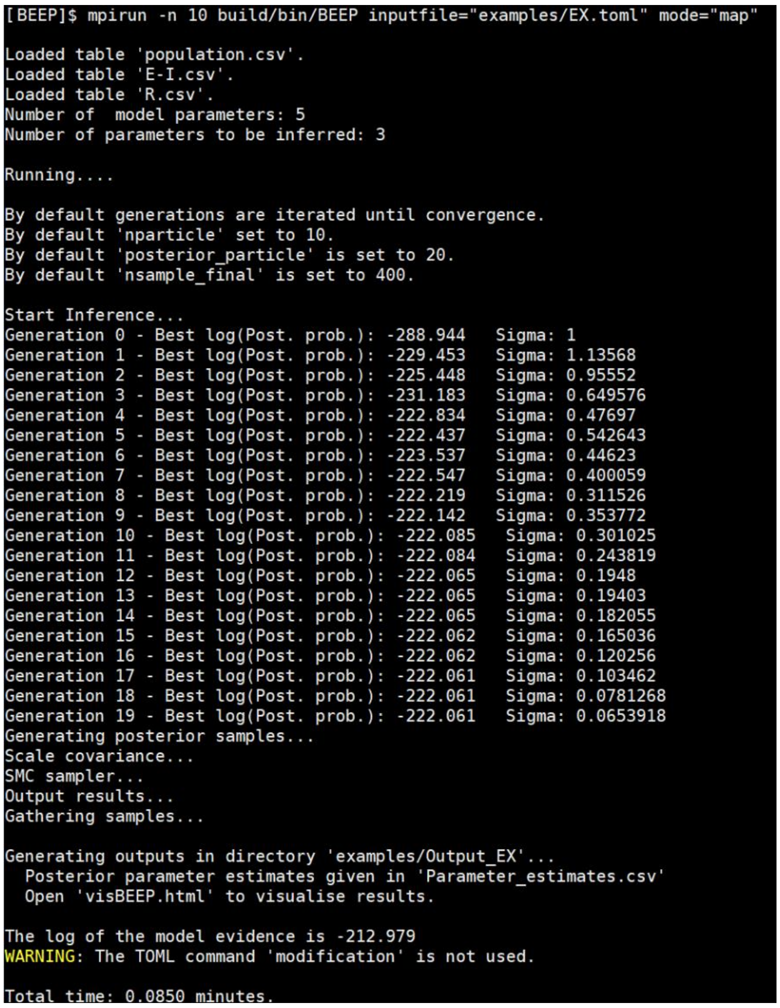
1.4.2 Inference example

We now look to perform inference on the simulated data files "E-I.csv" and "R.csv" generated in the previous section (note, these files have been placed in the "Data_EX" directory, but if BEEP can't find them there it automatically searches the "Simulated_data" folder 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 for 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 the "Parameter_estimates.csv" file in Fig. 8 we see that BEEP outputs posterior estimates for all model parameters.



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

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 10.
By default 'posterior_particle' is set to 20.
By default 'nsample_final' is set to 400.

Start Inference...
Generation 0 - Best log(Post. prob.): -288.944   Sigma: 1
Generation 1 - Best log(Post. prob.): -229.453   Sigma: 1.13568
Generation 2 - Best log(Post. prob.): -225.448   Sigma: 0.95552
Generation 3 - Best log(Post. prob.): -231.183   Sigma: 0.649576
Generation 4 - Best log(Post. prob.): -222.834   Sigma: 0.47697
Generation 5 - Best log(Post. prob.): -222.437   Sigma: 0.542643
Generation 6 - Best log(Post. prob.): -223.537   Sigma: 0.44623
Generation 7 - Best log(Post. prob.): -222.547   Sigma: 0.400059
Generation 8 - Best log(Post. prob.): -222.219   Sigma: 0.311526
Generation 9 - Best log(Post. prob.): -222.142   Sigma: 0.353772
Generation 10 - Best log(Post. prob.): -222.085   Sigma: 0.301025
Generation 11 - Best log(Post. prob.): -222.084   Sigma: 0.243819
Generation 12 - Best log(Post. prob.): -222.065   Sigma: 0.1948
Generation 13 - Best log(Post. prob.): -222.065   Sigma: 0.19403
Generation 14 - Best log(Post. prob.): -222.065   Sigma: 0.182055
Generation 15 - Best log(Post. prob.): -222.062   Sigma: 0.165036
Generation 16 - Best log(Post. prob.): -222.062   Sigma: 0.120256
Generation 17 - Best log(Post. prob.): -222.061   Sigma: 0.103462
Generation 18 - Best log(Post. prob.): -222.061   Sigma: 0.0781268
Generation 19 - Best log(Post. prob.): -222.061   Sigma: 0.0653918
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.

The log of the model evidence is -212.979
WARNING: The TOML command 'modification' is not used.

Total time: 0.0850 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 then loaded, providing information about weekly cases and the overall recovered population. Inference is performed. What is displayed here 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)). An estimate for the model evidence is provided which can be compared against alternative models to select which fits the data the best.

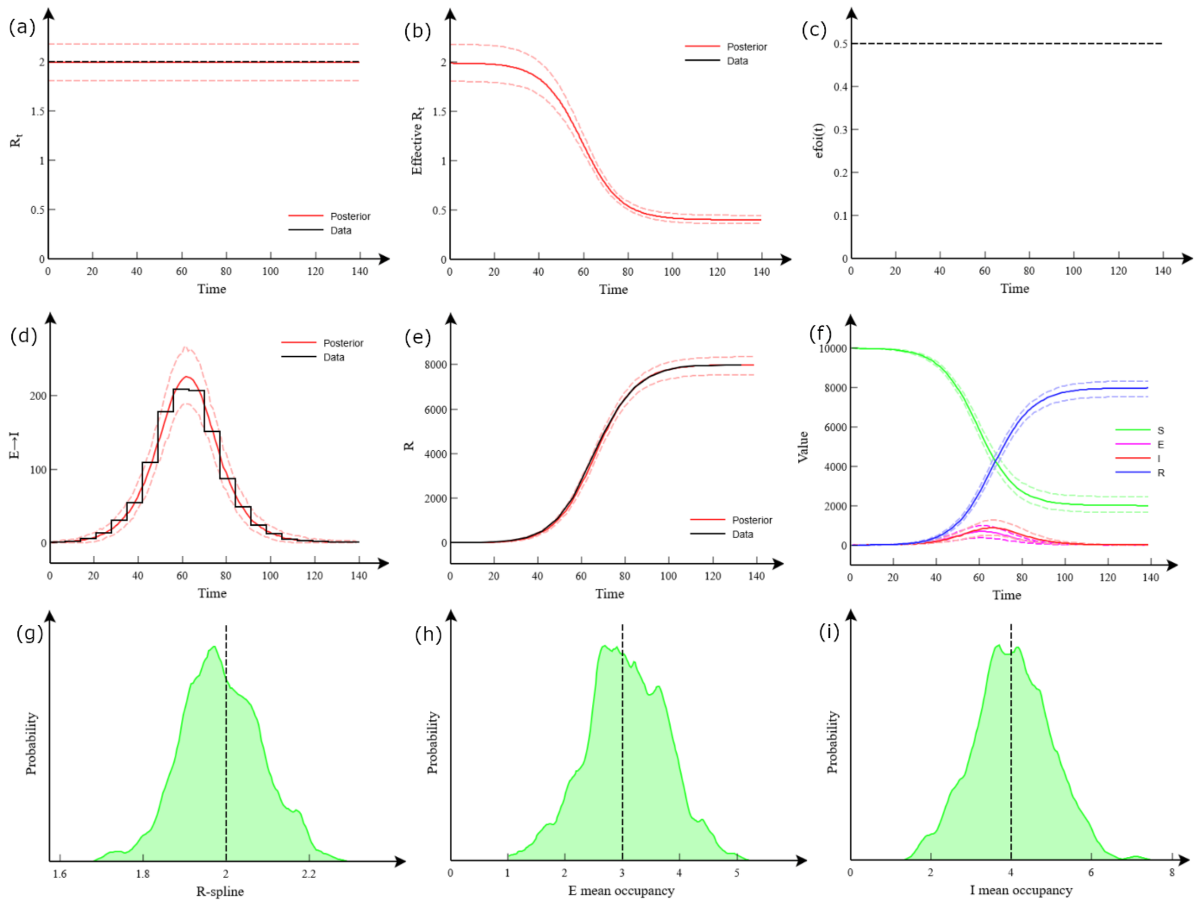
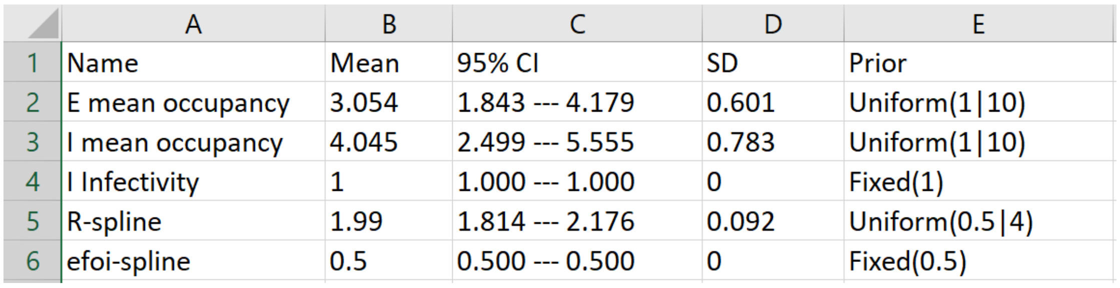


Figure 7. Inference from model. This shows some of the outputs from BEEP when performing inference using the data files in Fig. 5. (a) The posterior distribution for the reproduction number R_t, with the dashed lines representing 95% credible intervals. (b) The effective reproduction number R_t^{eff}. (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. 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 but having little impact once it has started).

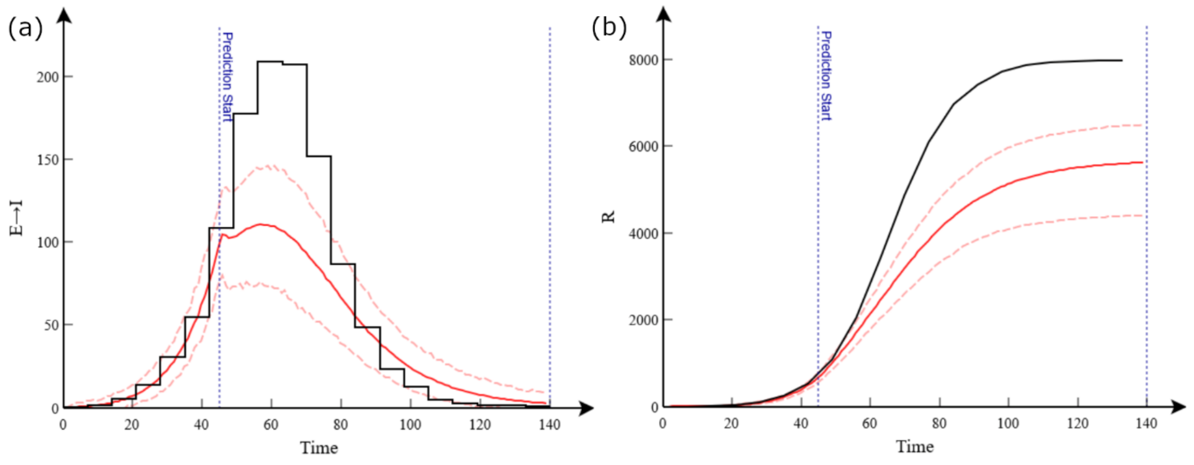


Figure 9. Prediction. These plots predict the effect of modifying the model such that the transmission rate is reduced by 30% after day 45, *e.g.* mimicking the impact of a lockdown. (a) The red curve gives the modified posterior distribution for the case rate. After the prediction start time there is 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 for making predictions:

- Type the following to run a 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") 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 – Index c is used to denote different compartments that specify an individual's disease status (*e.g.* in Fig. 1 index c can either be S, E, I or R). BEEP allows the user to specify an arbitrarily number of compartments in the model. For example, 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 – Index 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 (reflecting the fact that disease may affect groups in differing ways).

Geographic stratification – Index 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, 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 $\mathrm{S} \rightarrow \mathrm{E}$ transition). In the general case, the 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:

[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 peaked around its mean with an almost normal distribution.

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

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, 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_t^{eff}. 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, ψ=1, and for more than one strain, ψ_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:

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

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

$$\sum_a w_a \phi_a = 1, \tag{5}$$

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 $\mathbf{X}$ is a design matrix and $\boldsymbol{b}$ 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[5]. For example, suppose the first column in $\mathbf{X}$ gives the log of the local population density p_a. This would mean that Eq.(4) contains a factor proportional to

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

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 (*e.g.* relating disease transmission to time varying climatic data in different areas).

The last term in Eq.(4) 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

$$\sum_l h_l \phi_l = 1, \tag{7}$$

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:

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

[5] The quantity $\langle X_j \rangle$ is the average of $X_{a,j}$ over all areas. Incorporating this ensures only variation in the covariate, rather than its absolute value is used to correlate with disease transmission (so a global shift of all values within4 a column of the design matrix makes no different to the model).

where $Z_{a,a'}$ is a constant matrix (which is an input into BEEP and, for example, could be based on census commuter data[6], 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, corresponding to a frequency dependent model). 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 between areas and m_t=1 corresponds to mixing according to $Z_{a,a'}$). If movement restrictions 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 contact patterns in humans). In this case

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

where z and z' are the age groups corresponding to 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 $C_{z,z'}$ (*e.g.* which can be informed by previously published surveys such as POLYMOD [3] or the BBC Pandemic study [4], see Appendix D for details). However, time dependency can also be accounted for by defining modifications to this matrix. Specifically factors, defined by time-varying splines $f_{z,t}$, can we setup to multiply the rows and columns in $C_{z,z'}$:

$$B_{z,z',t} = f_{z,t} C_{z,z'} f_{z',t} \tag{10}$$

At the spline breakpoint the factors are constrained by

$$\sum_z f_{z,t} \phi_z = 1, \tag{11}$$

where ϕ_z is the proportion of the population in age group z (note, such a restriction is necessary otherwise the model becomes ill-defined). Equation (10) allows the model to capture how the mixing between different age groups changes over time with reference to the pre-pandemic estimate $C_{z,z'}$. This can be used in conjunction with age stratified data to estimate the impact of closing school, restricting access in care-homes, etc…, all of which have important policy implications.

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 – $N^c_{a',d',t}$ gives the total number of individuals in compartment c, area a' and demographic group d' at time t[7].

External infections – $\eta_{s,a,d,t}$ 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

[6] It is important to note that we only need to know the input $Z_{a,a'}$ up to a constant factor.
[7] Note, the demographic group d specifies the strain these individuals are infected with.

implementation). Note, here η is divided by the total population size N such that it represents the total mean number of external infections generated per unit time in the system as a whole.

2.1.3 System dynamics

BEEP uses a discrete time τ-leaping algorithm (with fixed time step τ) to approximate a continuous time Gillespie algorithm[8]. 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[9] 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 sampled from a Poisson distribution

$$n_{a,d,j_{\text{inf}},t} \sim Poisson\left(\tau p_{a,d_{-s},S,t} \lambda_{s,a,d,t}\right), \tag{12}$$

where S is the susceptible compartment[10], 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 is determined only after they become infected).

For non-infection transitions j, the number is sampled from

$$n_{a,d,j,t} \sim Poisson\left(\tau p_{a,d,i_j,t}(b_{d,j} / \mu_{d,i_j})\right), \tag{13}$$

where $b_{d,j}$ is the branching probability for an individual 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 Eqs.(12) and (13) populations needs to be updated:

$$p_{a,d,c,t+\tau} = p_{a,d,c,t} + \left[\sum_{j \in \text{enter c}} n_{a,d,j,t}\right] - \left[\sum_{j \in \text{leave c}} n_{a,d,j,t}\right], \tag{14}$$

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 μ are constructed by sequentially linking together k compartments, each with exponentially distributed residency time that has a mean of μ/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 S compartment[11]. Population stratification across different areas a and demographic groups d_{-s} are inputs which must be specified. For the most part these are

[8] τ can be specified, but by default it is set to 0.5 days.
[9] The possibility of individuals being infected with multiple strains is ignored.
[10] Here S is defined to be the state from which infection events occur, *e.g.* in Fig. 1 this corresponds to the "S" compartment.
[11] Alternatively the initial state can be specified in BEEP (see §3.6.2).

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 the 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 population sizes (and so, for example, may be sufficiently good to run a nationwide level analysis). In this case, the number of transitions is simply set to the mean of the Poisson distribution rather than being sampled from it:

$$\begin{aligned} n_{a,d,j_{\text{inf}},t} &= \tau p_{a,d_{-s},S,t} \lambda_{s,a,d,t}, \\ n_{a,d,j,t} &= \tau p_{a,d,i_j,t} b_{d,j} / \mu_{d,i_j}. \end{aligned} \tag{15}$$

2.2 Data

BEEP can accept data in the following types:

- **Time series transition data** – As shown by the file in Fig. 5(a), this is represented by a list giving the numbers 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.* with different columns representing different geographical areas). This can be used to incorporate, *e.g.*, case data, hospital admissions or death data.
- **Time series population data** – This gives the population in a compartment (or compartments) at a series of time points[12]. Again, this data can be stratified by age or demographic group, and can be used, *e.g.*, for the total number of people in hospital.
- **Time series population fraction data** – The same as population data except expressed as a fraction of the entire population. This can be used, *e.g.*, for seroprevalence data[13].
- **Marginal data** – This gives 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, which are referred to as ξ. In reality ξ is usually unknown and from a Bayesian point of view is considered a set of latent model variables. The data is collectively referred to as y.

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

$$\pi(\theta,\xi \mid y) \propto \pi\left(y \mid \xi\right) L(\xi \mid \theta) \pi(\theta), \tag{16}$$

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

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 eight different inference algorithms which aim to generate samples from the posterior distribution in Eq.(16). 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 a set of parameter values θ that maximise the *a posteriori* probability. The overall likelihood is found by integrating out the latent event space:

$$L(y|\theta) = \int \pi(y|\xi) L(\xi|\theta) d\xi. \tag{17}$$

Although it is not possible to evaluate this integral exactly, it can be calculated approximately. This rests on the assumption that, at any given point in time, the compartmental populations in the ensemble of possible stochastic outcomes from the system are multivariate normally distributed (MVN). Based on central limit theorem this approximation is valid when the number of infected individuals is large. The mean and covariance matrix for the MVN population distribution evolves in time according to a deterministic set of equations (see Appendix F for details). Observations on the system also act to modify the MVN, again in a deterministic manner. Integration of these equations over the time period under observation yields an estimate for $L(y|\theta)$.

Covariance matrix adaptation evolution strategy (CMA-ES) – This algorithm [5] 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.* it contains many meta-stable state). It starts with some initial guess for the parameters θ (*e.g.* taken from the prior) and proceeds over a series of generations to maximise $L(y|\theta)\pi(\theta)$ (*i.e.* converges to the MAP). CMA-ES also provides an approximation for the Hessian matrix, and so allows for estimation for parameter uncertainties by locally fitting a MVN approximation to the posterior around the MAP.

Whilst MAP offers an exact solution using an approximate posterior estimate, the subsequent methodologies look at an approximate solution to an exact posterior.

2.3.2 Power posterior

Although Eq.(16) represent an ideal, often it is found that generating true posterior samples takes an unfeasibly long CPU time, especially when y contains a lot of data[14]. One way around this is to consider a modified distribution called the power posterior [6]:

$$\pi_{PP}(\theta,\xi | y,\phi_{post}) \propto \left[\pi(y|\xi)\right]^{\phi_{post}} L(\xi|\theta)\pi(\theta), \tag{18}$$

where ϕ_{post} is the so-called inverse temperature. When ϕ_{post}=1 the true posterior is recovered and when ϕ_{post}=0 the system maps out the prior (with disregard for the data). Setting ϕ_{post} less than one

[14] 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.

can substantial speed up inference, but usually only marginally affects posterior estimates (essentially justifying this approach).

Four different inference algorithms aim to draw samples from the power posterior in Eq.(18):

Markov chain Monte Carlo with model-based proposals (MCMC-MBP) – This approach runs an MCMC chain at a specified ϕ_{post} using "model-based proposals" (MBPs) [7, 8]. MBPs allow for joint proposals in θ and ξ to explore the power posterior more efficiently than changes made in conventional data augmentation schemes[15].

Metropolis coupled MCMC (MC^3) – Here K separate MCMC chains, indexed by k, are run in parallel (again, making use of MBPs) [9]. The inverse temperatures of these chains are selected to span from the posterior (k=1) to the prior (k=K) [16] according to:

$$\phi_k = \left(\tfrac{K-k}{K-1}\right)^4 \phi_{post}. \tag{19}$$

This approach has the advantage of improved mixing[17] (states can be swapped between chains), it 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 additional computational burden of running multiple chains making it slower than some other approaches.

Particle anneal sampling using MBPs (PAS-MBP) - PAS-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 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 [10]. 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 big datasets 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 using the full observation model in Eq.(16), approximate Bayesian computation (ABC) [11] relies on introducing a measure of fit between the data y and the system state ξ called the error

[15] These would typically separately make small changes to each of the quantities in $n_{a,d,j,t}$ or each of the model parameters in θ and accept or reject these changes based on a Metropolis-Hastings probability.
[16] Note, the power 4 is found to empirically work well under many scenarios. This factor can be specified using the 'invT_power' command.
[17] Good mixing implies that consecutive samples along an MCMC chain are not highly correlated.

function (EF). For convenience we assume that the error function is related to the observation model (see the next section) through[18]

$$EF(y\,|\,\xi) = -2\log\left(\pi\left(y\,|\,\xi\right)\right). \tag{20}$$

Because of the minus sign this means that a higher observation probability corresponds to a lower error function and *vice-versa*[19].

We define a model "sample" to be generated by first sampling a set of model parameters θ from the prior $\pi(\theta)$ and then simulating a state ξ from the model. Under ABC the posterior distribution is approximated by sampled states θ and ξ for which the EF is below a given cut-off value EF_{cutoff}. This can be represented by

$$\pi_{ABC}(\theta,\xi\,|\,y,EF_{\text{cutoff}}) \propto H\left(EF(y\,|\,\xi) - EF_{\text{cutoff}}\right)L(\xi\,|\,\theta)\pi(\theta), \tag{21}$$

where H is the Heaviside step function. BEEP incorporates three algorithms for generating approximate posterior samples using this distribution:

Approximate Bayesian Computation (ABC) – This generates posterior samples by means of the simple ABC rejection sampling scheme [11]. Samples from the model are generated (as described above) and only those which have an error function below the cut-off are accepted. This approach becomes highly inefficient as the cut-off threshold is reduced because the vast majority of sampled states are discarded.

Approximate Bayesian Computation using sequential Monte Carlo (ABC-SMC) – This runs over a series of generations, with the previous generation being used as an importance sampler for the next [12]. This approach is usually significantly more computationally efficient than the standard ABC rejection sampling algorithm.

Approximate Bayesian Computation with model-based proposals (ABC-MBP) – This starts with a number of randomly sampled "particles" from the prior (where here a particle comprises of a parameter set θ and a system state ξ). The algorithm proceeds through a series of generations and within each generation: 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 can be used in BEEP to relate the observed data to equivalent quantities derived from the state ξ (note through the relationship in Eq.(20) this also defines potential error functions that can be used).

For simplicity BEEP assumes that observations are independent, and so

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

[19] Note, however, that unusually an exact agreement with the data and the state does not imply a zero error function (as it would do in most ABC schemes). However, this arbitrary shift in the origin is not problematic because it has no effect on inference.

$$\pi(y|\xi) = \prod_i \pi(y_i | Y_i(\xi)), \tag{22}$$

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. 5) 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:

  $$\pi(y_i | Y_i(\xi)) = N(y_i | Y_i(\xi),_i \sigma_i), \tag{23}$$

  The value of the standard deviation σ_i can either be set separately for each observation, defined as a specific value for a given data source, or as a percentage of the observation itself. In the latter case the following relationship is used:

  $$\sigma_i = 0.01 \times p_i \left(y_i(1-\varepsilon) + y_{\max}\varepsilon \right), \tag{24}$$

  where p_i is the percentage, y_i is the observed value, y_{max} is the maximum observed value for a given time series, and ε is a small constant (set to 0.01 by default) which ensures σ_i does not become zero when y_i becomes zero.

- **Poisson** – The data is Poisson distributed with mean given by the state value:

  $$\pi(y_i | Y_i(\xi)) = Poisson(y_i | Y_i(\xi)). \tag{25}$$

- **Negative binomial** – The data has a negative binomial distribution around the state value:

  $$\pi(y_i | Y_i(\xi)) = NB(y_i | Y_i(\xi), k), \tag{26}$$

  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 μ).

The options above can be selected separately for each of the data files loaded into BEEP. Incorporation of thresholds into the model (*i.e.* data which indicates that a value is below a given threshold[20]) is discussed in Appendix E.

2.3.5 Incomplete observations

Some measureable quantities directly fit into the data types specified in §2.2, *e.g.* statistics on hospitalisations and deaths. For other measureable quantities, however, things can 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

[20] This is sometimes used to ensure non-identifiably of individuals.

tests are imperfect and do not pick up all infections)[21]. 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 directly compare to the observed data y_i. In particular a time-dependent scaling can be added:

$$Y_i' = s_t Y_i, \tag{27}$$

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.

2.3.6 Model evidence

The model evidence (ME) $\pi(y)$ provides a measure for comparing how well a model fits the data y[22]. Selecting the highest ME from a range of models allows the user to select which supports the data the best. In particular, the ratio in ME between two models gives the Bayes factor (a factor above 3.2 is considered substantial support for one model over another, and over 10 provides strong evidence [13]).

For the MAP approach introduced in §2.3.1 the ME can be estimated directly. First the Hessian matrix $\mathbf{H}$ is calculated, which gives the double derivative of the log of the posterior probability distribution. The model evidence is then approximated using Laplace's method as:

$$\pi(y) \simeq \frac{(2\pi)^{k/2}}{\sqrt{|\mathrm{H}|}} L\left(y \mid \theta_{MAP}\right) \pi\left(\theta_{MAP}\right), \tag{28}$$

where k is the number of model parameters, θ_{MAP} is the MAP estimate, $L(y|\theta_{MAP})$ is the data likelihood introduced in Eq.(17) and $\pi(\theta_{MAP})$ is the prior. Here the likelihood term gives the probability of observing the data given θ_{MAP} is the true parameter set. If this is high the model evidence will also be high, as would be expected. The other two terms in Eq.(28) penalise against models with greater complexity (avoiding overfitting).

In approaches which explicitly incorporate the latent system state ξ, one way to define the ME is as the expected value of the observation model over random samples from the model[23]:

$$\pi(y) = \left\langle \pi\left(y \mid \xi\right) \right\rangle_{\xi} \tag{29}$$

Models that generate samples in good agreement with the data have a higher ME (because their observation model probability will be higher). As discussed above, for most practical applications it is not possible to sample from the true posterior. Following §2.3.2, the ME for the power posterior is given by

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

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

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

$$\pi(y\,|\,\phi_{post}) = \left\langle \left[\pi\left(y\,|\,\xi\right) \right]^{\phi_{post}} \right\rangle_{\xi}, \tag{30}$$

and for ABC approaches in §2.3.3:

$$\pi(y\,|\,EF_{\text{cutoff}}) = \left\langle H\left(EF(y\,|\,\xi) - EF_{\text{cutoff}} \right) \right\rangle_{\xi}. \tag{31}$$

It is important to note that in the case of Eqs. (30) and (31) the ME is not just a single value, but depends on either the inverse temperature ϕ_{post} or the 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}).

Appendix G describes how the ME is calculated for those algorithms which can estimate this quantity[24].

2.3.7 Which inference algorithm should I use?

This section has introduced a bewildering array of inference algorithms implemented in BEEP. For most users the key question will be, which should I use? For most models tested (see §6) MAP using CMA-ES is found to work well at providing unbiased estimates for model parameters and good approximations for credible intervals. It is also generally the fastest approach, so it would be recommended as the approach to try first. It is worth bearing in mind, however, that the following restrictions apply: 1) Populations must be sufficiently large such that the MVN compartmental population approximation is valid, 2) the observation model is restricted to be normal, 3) when transition data is used the compartmental model cannot contain backward transitions[25], and 4) the posterior marginalised in parameter space is also assumed to be MVN. Of the other approaches ABC-MBP using an ABC approach and PAS-MBP using a power posterior approach are found to run the fastest.

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

[25] *E.g.*, waning immunity cannot be incorporated because it relies on a transition back to the susceptible compartment.

3 Inputs into BEEP

BEEP requires an input TOML file in order to define the model (§2.1), reference data files (§2.2), and perform simulation, inference (§2.3) or prediction. For reference a glossary of all TOML commands is given at the end of this manual. BEEP also contains numerous examples in §6 which help to illustrate practical uses for this software.

3.1 A simple example

We first go through the following simple example (taken from the file "examples/EX.toml" and first introduced in §1.4) explaining what each of the TOML commands does:

```
time_format = "number"
start = "0"
end = "140"
datadir = "examples/Data_EX"
outputdir = "examples/Output_EX"
comps = [{name="S"},
         {name="E", dist="Erlang", k="2", mean_value="3", mean_prior="Uniform(1,10)"},
         {name="I", dist="Exp", mean_value="4", inf_value="1", mean_prior="Uniform(1,10)"},
         {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 the analysis.

'datadir' defines the location of the data directory.

‘outputdir’ indicates where outputs from BEEP are placed.

‘comps’ defines 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 an exponential distribution and "Erlang" denotes an 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[26]. The relative infectiousness is set by ‘inf_value’, and if omitted is assumed zero[27].

‘trans’ defines 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 $\eta_{s,a,d,t}$ in Eq.(1). This is used to induce infections in the system that initiate epidemics in the first place. In this case it is set to the small value of 0.5 to initiate an epidemic.

‘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). In this case the file “population.csv” contains the following information:

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

One column in this file must have the heading “area”, and this refers to the geographical area under consideration. This particular example is for 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 models). 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 week from the file “E-I.csv” and the population in the R compartment at weekly intervals from the file “R.csv” (note, when run in inference mode these files should be placed in the data directory, whereas in simulation mode these data files are created in the “Simulated_data” sub-directory in the output folder). Figure 5 shows these data files. Under inference ‘obsmodel’ specifies the

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

observation model used to relate the true system state values to observed data values (in this example a normal observation model is used with a 10% standard deviation).

‘state_outputs’ can be used to generate additional graphical plots (*e.g.* see Fig. 4(f) and Fig. 7(f)).

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

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 any specified parameter can have a prior distribution associated with it and this is used when inference is performed. Three types of prior 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 specified 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:

- "Dir(*)" is used to apply an uninformative Dirichlet flat prior.
- Dir(α) applies an informative prior with a specified value for α. If b_j represents the probability of passing down branch j, the overall probability distribution for all branches is represented by

$$\frac{\Gamma(\sum_j \alpha_j)}{\sum_j \Gamma(\alpha_j)} \prod_j b_j^{\alpha_j - 1}, \tag{32}$$

  where Γ is the gamma function. If all α_j are set to one this becomes a flat prior.
- "Dir(μ,σ)" automatically sets α_j to give an informative prior with a specified mean μ and standard deviation σ[28].

Modified Dirichlet priors – These are used is cases in which the weighted average of a quantity is required to be one, *i.e.* relative susceptibilities, area effects, and level effects[29]. For example, area effects in §6 EX C1 are applied using:

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

"MDir(σ)" applies an informative prior with a specified standard deviation parameter σ. Suppose the population is divided into A groups, indexed by a, each with a population fraction ϕ_a and value v_a. The

[28] 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.
[29] This ensures the normalisation conditions in Eq.(3), (5) and (7) are strictly satisfied.

weighted average $\sum_a v_a \phi_a$ must be one. To ensure this, each terms $v_a \phi_a$ is set to have a Dirichlet prior Dir(α_a) with

$$\alpha_a = \left(\frac{A-1}{\sigma^2} - 1 \right) \phi_a. \tag{33}$$

This choice leads to the prior for v_a having a mean of one and a variance

$$\sigma^2 \frac{\frac{1}{\phi_a} - 1}{A-1}. \tag{34}$$

Note, when all groups have equal population fraction, *i.e.* $\phi_a = 1/A$, this simplifies to a variance of σ^2 for all (or equivalently a standard deviation of σ). In the general case those groups with a smaller population fraction have a larger variance than average. The choice "MDir(0.5)" from above represents a relatively uninformative prior choice (allowing for a large ~50% variation).

Ordering priors – Priors can be set that enforce the ordering of certain parameters. 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[30]. 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 for priors to be as restrictive on model parameters as possible.

3.3 Splines

Piecewise linear splines are used in BEEP to capture time variation in model parameters (see Fig. 10). We next provide a description of how these splines are defined (note, the examples below use the spline for the reproduction number 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 it is assumed the quantity remains constant over time. In the case above, intermediate breakpoints are set on March 1st and July 15th 2020.

[30] In which case the posterior distribution almost maps out the prior.

‘param’ defines the names of parameters associated with each of the breakpoints. If ‘param’ is unspecified, 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.

.

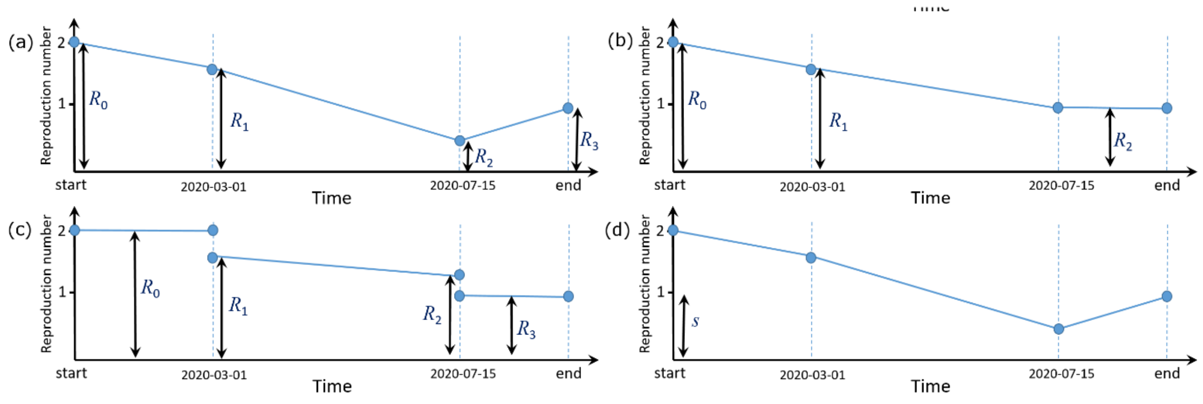


Figure 10: Types of spline – This shows different ways in which splines can be defined (see §3.3 for details). (a) A piecewise linear spline defined by four breakpoints with four parameters R_0-R_3 specifying the heights at each breakpoint. (b) The last two breakpoints share the same parameter (hence 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 prior specification:

```
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 onto the spline itself (which can be 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[31]. Three different types of smoothing can be selected from:

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

$$\pi(\theta^{spline}) = \prod_{b=1}^{B-1} N(\theta_{b+1}^{spline} \mid \theta_{b}^{spline}, s^2), \tag{35}$$

  where N is the normal probability distribution, b indexes the B breakpoints and θ_b^{spline} represents the model parameters at the breakpoints.
- "log_smooth" – Implies a smoothing prior given by

$$\pi(\theta^{spline}) = \prod_{b=1}^{B-1} LN(\theta_{b+1}^{spline} \mid \theta_{b}^{spline}, s^2), \tag{36}$$

  where LN is the log-normal probability distribution. 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[32]:

```
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",
            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.(35) and (36).

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).

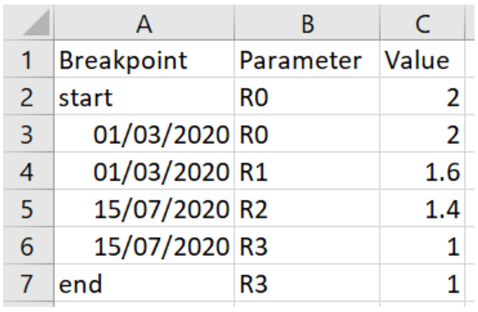
[31] This strength can be set individually for each breakpoint by separating values using the ‘|’ character.
[32] In this case, the numbers set in ‘values’ must be the same for repeated use of the same parameter.

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]".



	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.

3.3.5 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.

3.4 Setting up a 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 'comps' 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 the mean 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 'comps', an age-based dependency on the infectious period could be added in the following way:

```
 {name="I", dist="Exp", mean_param="age: I_Y | I_A | I_E",
                              mean_value="age: 2.6 | 3.4 | 5", inf_value="1"}
```

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

```
 { name="I", dist="Exp", mean_param=" age,sex: 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", inf_value="1"}
```

would be used to specify both 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'[33]. In this example, a different spline is fit to England and Scotland[34]:

```
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 for external infections[35]. 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

From an epidemiological point of view different demographic groups in the population may behave differently. This section described how these groupings can be specified.

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).

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).

[33] In essence R_t is replaced by $R_{a,t}$ in Eq.(1).

[34] Note, from §3.6 it is assumed that the 'areas' file contains a column with the heading "Nation" and "England" or "Scotland" are specified on different rows.

[35] This is because once local transmission gets going the external contribution to the overall infection rate is typically very low.

Optionally, BEEP also allows for modification of the age mixing matrix by time varying factors, as shown in Eq.(10). These are incorporated through the ‘age_mixing_perturb’ command. For an example taken from §6 EX E2:

```
age_mixing_perturb = [
 { agecat='age0-19', bp='[bp:amp.csv]', param='[Par0-19:amp.csv]', value='[Value0-19:amp.csv]',
                              prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'},
 { agecat='age20-39', bp='[bp:amp.csv], param='[Par20-39:amp.csv]', value='[Value20-39:amp.csv]',
                              prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'},
 { agecat='age40-59', bp='[bp:amp.csv]', param='[Par40-59:amp.csv]', value='[Value40-59:amp.csv]',
                              prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'},
 { agecat='age60+', bp='[bp:amp.csv]', param='[Par60+:amp.csv]', value='[Value60+:amp.csv]',
                             prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'}
 ]
```

This model contains four age groups 0-19, 20-39, 40-59 and 60+.

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 overall number of compartments in the model is multiplied by the number of categories in each classification, so the CPU time for inference substantially increases when more classifications are incorporated into the 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’ takes a modified Dirichlet prior.

Similar variations in susceptibility can be applied to other demographic categories via the ‘democats’ command.

3.5.4 Incorporating changes in demographic groups

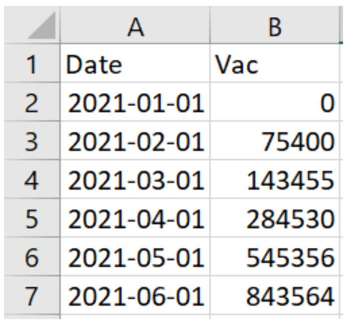
The command ‘democat_change’ allows for the proportions within a specified demographic classification in the susceptible population to vary over time (*e.g.* this can be used to study the effect of vaccination), for example

```
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’ or “Time” 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.



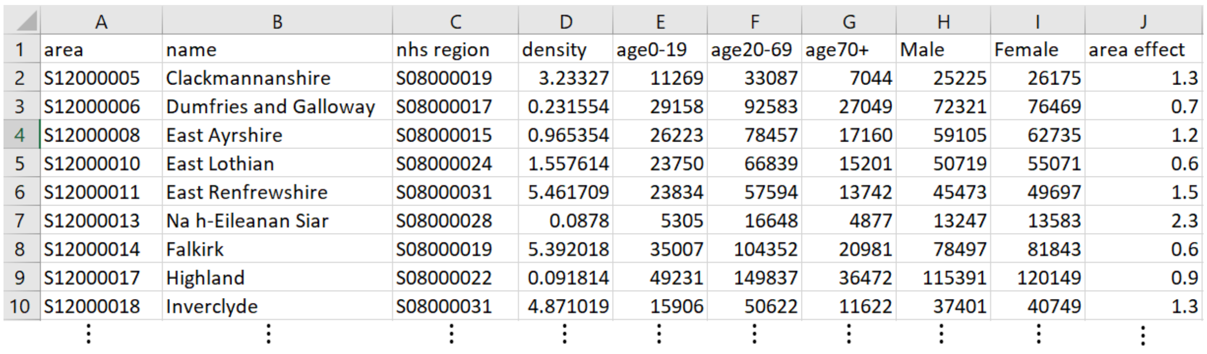
	A	B
1	Date	Vac
2	2021-01-01	0
3	2021-02-01	75400
4	2021-03-01	143455
5	2021-04-01	284530
6	2021-05-01	545356
7	2021-06-01	843564

Figure 12. Vaccination data. Shows example file “vac.csv”.

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	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 file specifying different local authorities in Scotland (taken from “example/Data_EX_C5/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 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, in §6 EX C5

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

defines information about local authorities in Scotland (an extract of this file is shown in Fig. 13).

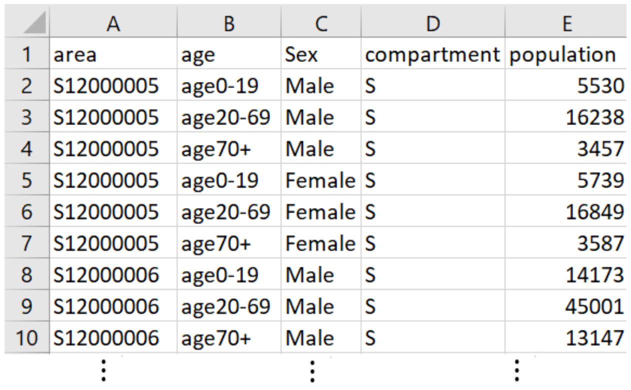
Generally speaking, the ‘areas’ file must contains the following columns: “area” (which gives a code or name that uniquely identifies each area), a specification of the initial population make-up (see below) and columns for any covariates in the model (*e.g.* population density)[36]. Optionally, other coarser geographical scales can be specified and reference in the model/data (*e.g.* in Fig. 13 'nhs region' amalgamates local authorities into NHS regions and this could be used if hospital admissions data was available on this geographical scale).

3.6.2 The initial population

The initial population ($p_{a,d_s,c,,t_{\text{start}}}$ in §2.1.3) is loaded into BEEP in one of two ways:

Within the file specified by 'areas' – If the model is not age stratified, a column with the heading "population" gives the total population size within each area. For age-stratified models, BEEP looks for column headings corresponding to the specified age categories (*e.g.* "age0-19", see Fig. 13) which specify the populations in each of these categories. For other demographic classifications the file must contain column headings corresponding to each category, with values given as either populations, percentages, or fractions[37].

For simplicity, it is assumed that the probability 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 reasonably valid in most cases, although it can break down (*e.g.* individuals in older age groups are predominantly female).



	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" giving the number of individuals. The file consists of a series of rows that list every possible combination of options followed by the corresponding population within that grouping (*e.g.* see Fig. 14).

3.6.3 Geographical mixing

When more than one area is specified it becomes necessary to load a matrix that captures the interactions of individuals within and between areas. This is implemented 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*-

[36] 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 in the analysis.
[37] BEEP automatically works out which of these possibilities is being used.

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 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. Note, the specified 'geo_mixing_matrix' only needs to be specified up to a constant factor.

Optionally, the spline m_t in Eq.(8), which allows for time variation in mixing, can be 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 splines). If unspecified, m_t=1 is assumed.

3.6.4 Area effects

These account for the fact that disease transmission may vary across different areas (indexed by a). Following the four contributions to area effects in Eq.(4):

Relative area effects w_a – These capture any residual variation not explainable by any other effects (discussed below). They are incorporated by

```
area_effect = {value="[area effect:Scotland_LA.csv]", prior="MDir(0.2)"}
```

Note, in Fig. 13 the column 'area effect' contains values used when simulating from the model. A modified Dirichlet prior is used when performing inference (which ensures an average relative area effect of 1).

Fixed effects b_j – These relate transmission in an area to properties of that area (*e.g.* population density). Columns of the design matrix $\mathbf{X}$ 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 either be "linear" to leave it unchanged, or "log" to log transform). This example shows one fixed effect but more columns in $\mathbf{X}$ can be added by comma separating terms within the square brackets.

Time-varying fixed effects $\boldsymbol{b'_j}$ – These relate transmission in an area to time-varying properties of that area (*e.g.* temperature). 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). This example shows one time-varying fixed effect, but more can be added by comma separating terms within the square brackets.

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

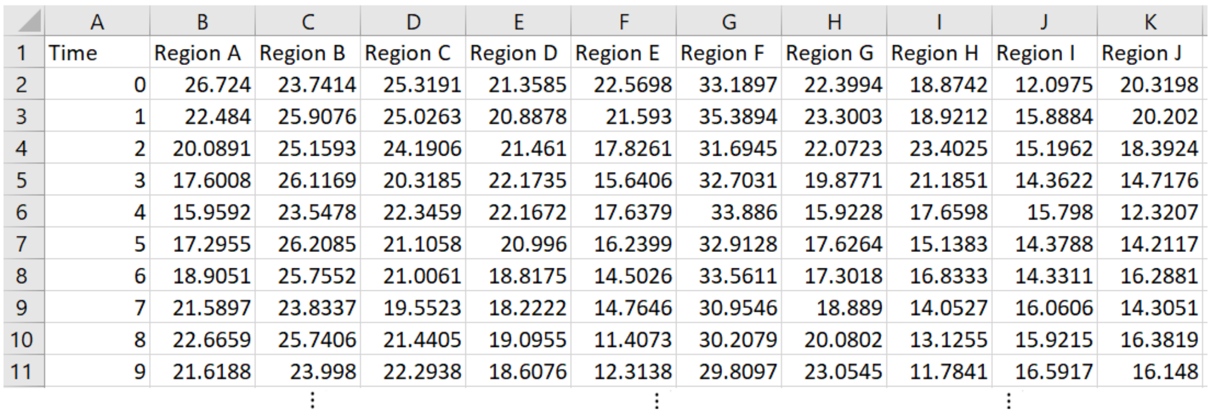
```
tv_covars = [{name="covar", file="tvcovars.csv", param="b2", value="0.1",
                   prior="Uniform(0,1)", func="linear"}]
```

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

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

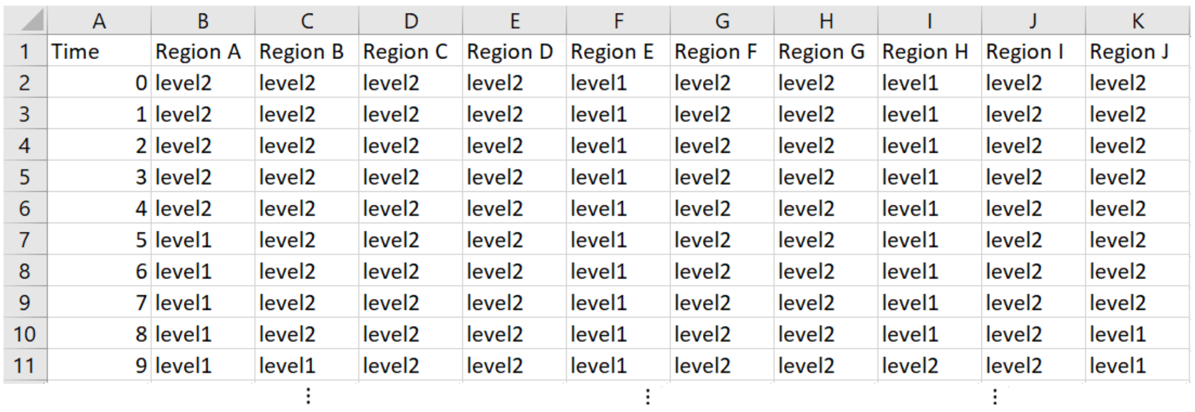
```
level_effect = {file="levels.csv", param="level1 | level2", value="0.5 | *", prior="MDir(0.5)"}
```

The file ‘levels.csv’ contains a table with dates/times going down rows and areas along columns (see Fig 16 for an example). Level effects are required to have an weighted average value of one, and so their prior takes a modified Dirichlet distribution.



	A	B	C	D	E	F	G	H	I	J	K
1	Time	Region A	Region B	Region C	Region D	Region E	Region F	Region G	Region H	Region I	Region J
2	0	26.724	23.7414	25.3191	21.3585	22.5698	33.1897	22.3994	18.8742	12.0975	20.3198
3	1	22.484	25.9076	25.0263	20.8878	21.593	35.3894	23.3003	18.9212	15.8884	20.202
4	2	20.0891	25.1593	24.1906	21.461	17.8261	31.6945	22.0723	23.4025	15.1962	18.3924
5	3	17.6008	26.1169	20.3185	22.1735	15.6406	32.7031	19.8771	21.1851	14.3622	14.7176
6	4	15.9592	23.5478	22.3459	22.1672	17.6379	33.886	15.9228	17.6598	15.798	12.3207
7	5	17.2955	26.2085	21.1058	20.996	16.2399	32.9128	17.6264	15.1383	14.3788	14.2117
8	6	18.9051	25.7552	21.0061	18.8175	14.5026	33.5611	17.3018	16.8333	14.3311	16.2881
9	7	21.5897	23.8337	19.5523	18.2222	14.7646	30.9546	18.889	14.0527	16.0606	14.3051
10	8	22.6659	25.7406	21.4405	19.0955	11.4073	30.2079	20.0802	13.1255	15.9215	16.3819
11	9	21.6188	23.998	22.2938	18.6076	12.3138	29.8097	23.0545	11.7841	16.5917	16.148
			⋮			⋮				⋮	

Figure 15: Time-varying fixed effects. This shows an extract from ‘examples/Data_EX_D2/areatvcovars.csv’, which is a fictional dataset giving daily air temperature measurements. Different columns represent ten different areas.



	A	B	C	D	E	F	G	H	I	J	K
1	Time	Region A	Region B	Region C	Region D	Region E	Region F	Region G	Region H	Region I	Region J
2	0	level2	level2	level2	level2	level1	level2	level2	level1	level2	level2
3	1	level2	level2	level2	level2	level1	level2	level2	level1	level2	level2
4	2	level2	level2	level2	level2	level1	level2	level2	level1	level2	level2
5	3	level2	level2	level2	level2	level1	level2	level2	level1	level2	level2
6	4	level2	level2	level2	level2	level1	level2	level2	level1	level2	level2
7	5	level1	level2	level2	level2	level1	level2	level2	level1	level2	level2
8	6	level1	level2	level2	level2	level1	level2	level2	level1	level2	level2
9	7	level1	level2	level2	level2	level1	level2	level2	level1	level2	level2
10	8	level1	level2	level2	level2	level1	level2	level2	level1	level2	level1
11	9	level1	level1	level2	level2	level1	level2	level2	level2	level2	level1
			⋮			⋮				⋮	

Figure 16: Level effects. This shows an extract from the file ‘examples/Data_EX_D3/levels.csv’. This fictional dataset assigns each area to one of two disease intervention levels as a function of time (on different days). Different columns represent ten geographical areas.

3.7 Data tables

This section describes how the command 'data_tables' is used to load data files into an analysis. 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 (see §2.2):

- "transition" – Time series information about the number of individuals moving down a specified transition(s).
- "population" – Time series population numbers within a specified compartment(s).
- "population_fraction" – Time series population fractions within a specified compartment(s).
- "marginal" – 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 name followed by "->" and the final compartment name, *e.g.* "E->I". Multiple transitions can be specified by comma separating, *e.g.* "I->D,I->R" would represent all individuals which cease to be infectious in an SIRD model. For "population" or "population_fraction" this is the compartment name under measurement, *e.g.* "R". Multiple compartments are comma separated, *e.g.* "E,I" would represent all infected individuals in an SEIR model.

'timestep' sets an integer giving the number of time units between observations (optional, by default set to one under simulation from the model).

'file' specifies the name of the file. This must either be 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" (in which case "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 of units (*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. If unspecified these are automatically taken from the data tables provided (for marginal data they are assumed 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"’, 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, they are 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 and strain dependency, ‘democats_dep="age"’ or ‘democats_dep="strain"’ are used, respectively.

‘democats_filt’ filters the demographic group observed (*e.g.* ‘democats_filt="Sex:Male"’ would mean that observations relate to 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 used to specify the character or string that denotes those missing value, *e.g.* ‘nodata_str = "NA"’ (by default this is set to ".").

For reasons of non-identifiablity, 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 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 to be transformed before it can be incorporated into the model (as discussed in §2.3.5):

‘factor’ multiplies the value given in ‘observation’ to make it comparable with the observed 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.(27)). The spline itself is specified by the ‘obs_spline’ command. For example

```
data_tables = [{type="transition", observation="E->I", file="E-I.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 to 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.* a relative 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 factors by which R_t is increased to account for differences in transmission rate for different strains (typically one strain is set to 1 to become the reference against which others are compared). This sets the parameters ψ_s in Eq.(1).

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

3.9 System dynamics

As discussed in §2.1.3, the system dynamics are approximated using a finite time step τ-leaping algorithm. The fixed value of 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 = 4
```

and by default has a value 2 (*i.e.* τ is half a day). Reducing Q speeds up inference but also can lead to substantial discretisation error. Crucially, τ should be significantly less than all the compartmental residency times specified in the model.

The underlying system dynamics can be set to deterministic (see Eq. (15)) using the command

```
dynamics = "deterministic"
```

or alternatively to "stochastic" (assumed by default)[38].

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 force of infection in the model is reduced by a factor of a half (*e.g.* mimicking the effect of a lockdown).

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

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

‘type’ denotes the type of change:

- "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).

‘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

If needed, it is possible to add other informative plots into the BEEP outputs (see §5). The command ‘state_outputs’ works in much the same way as ‘data_tables’, but rather than reading input files (or generating simulated data) here graphs are generated corresponding to particular observations of interest. This example plots the dynamic variation in compartmental population is an SEIR model:

```
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 multiple times ensures that lines are placed onto the same plot.

‘line_colour’ sets the colour of the line. This can take the values ‘black’, ‘red’, ‘blue’, ‘green’, ‘yellow’, ‘cyan’, ‘magenta’ (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 reached.

3.11.3 Probability distribution plots

BEEP allows for 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 with labels. These can be specified in the following way:

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

Multiple labels can be added by comma separating 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 at 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 parameters and then perform inference on those parameters to check if they are correctly identified. To place these simulated “true” values onto the output plots, 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. 7(d)). 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 it can be run by executing "build/bin/BEEP" on the command line followed by a specification of the input file name. Whilst the TOML file can, in principle, contain all the information BEEP needs to run an analysis, usually additional specifications are made on the command line to set the mode of operation and associated parameters[39] (this makes it easy to quickly change analysis or mode of operation 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
```

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'[40] (optional).

4.1.2 Multiple simulation

This mode simulates from the model multiple times and outputs the distribution in states generated:

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

Visualising the results allows the user to see the inherent stochastic variation in the model.

The following option can be specified:

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

4.2 Inference

BEEP incorporates eight different inference algorithms (briefly discussed in §2.3) which all aim to perform essentially the same task: to generate posterior samples for model parameters and the system state. They are split into three broad classes with different aims: 1) maximise an approximation to the *a posteriori* probability directly (MAP), 2) reduce an error function (ABC-MBP, ABC, ABC-SMC) and 3) incorporate a specified observation model, potentially smoothed with an inverse temperature (PAS-MBP, PMCMC, MCMC-MBP, MC^3). A discussion on which of these methods is the best to choose is given in §2.3.7.

4.2.1 Maximum *a posteriori* probability (MAP) using CMA-ES

The following command implements the CMA-ES algorithm:

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

with the following options:

[39] These will override any existing specifications in the TOML file itself.
[40] 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 (optional, by default set to 4+(5*log(*k*)) where k is the number of model parameters). If the system regularly becomes stuck in local optima (indicated by the algorithm converging on different solutions with different values of the seed), a larger particle number can potentially help (it allows CMA-ES to perform a more global search).
- *nsample* – Sets the number of posterior state samples (optional, by default set to 200). A larger number leads to a better approximation for state-based plots (e.g. Fig. 7(d)) and more accurate predictions (when BEEP is subsequently run in 'prediction' mode). Note, this number has no impact of the number of posterior parameter samples (*e.g.* used to generate parameter posterior distributions), which is fixed to 2000.
- *posterior_particle* – Sets the number of particles used in the sequential Monte-Carlo scheme that generates the final posterior state samples (optional, by default set to 20). Making this number larger improves the accuracy with which state samples are generated (but does not impact parameter estimates).

By default generations are iterated until there is no substantial change in the MAP estimate over the last 10 generations. Alternative termination conditions can be specified:

- *ngeneration* – Gives the number of generations.
- *cpu_time* – Iterates generations until a specified CPU time is reached (in minutes), at which point the algorithm terminates and outputs are plotted.

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:

- *nparticle* – Sets the number of particles (200 is typical as this yields approximately 100 random samples from the posterior).
- *ngeneration* – Sets the number of generations (a suitable value is problem dependent, but typically this lies in the range 10-100, depending on model complexity).
- *cpu_time* – As an alternative to setting '*ngeneration*', this iterates generations until a specified CPU time is reached (in minutes).
- *cutoff_final* – As an alternative to setting '*ngeneration*', this iterates generations until a specified EF_{cutoff} is reached.
- *cor_max* – Sets the maximum correlation in parameters between one generation and the next (by default set to 0.5). A lower value ensures that particles are less correlated at the end of each generation (leading to a larger effective particle number) but each generation takes longer to calculate. The value 0.5 sets an approximately optimal balance between these two competing effects.
- *cor_max_last* – Sets the maximum correlation in parameters between the penultimate and last generation (by default set to 0.5). If 'cor_max' is set to a large value (to speed up generation) 'cor_max_last' can be specified to ensure that particle in the last generation are uncorrelated.

4.2.3 Particle annealed sampling with model-based proposals (PAS-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="pas"  nparticle=200  ngeneration=40
```

This has the same 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* – 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 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 usually difficult in practice), but has the disadvantage of using much more memory (because of the necessity to store all generated states).

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

This runs over a series of generations, with the previous generation used as an importance sampler for the next. Under most 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* – Specifies the number of samples in each generation.
- *ngeneration* – Sets the number of generations.
- *cpu_time* – As an alternative to setting '*ngeneration'*, this iterates generations until a specified CPU time is reached (in minutes).
- *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 likelihood in Eq.(17) (which gives 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 – Sets 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* – Sets 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 iterations 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 chain 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* – Sets 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 iterations 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 (MC^3)

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=10  invT_final=1
```

with the following options:

- *nchain* – Sets the number of MCMC chains used (this needs to be sufficiently large to allow for adjacent chains to "overlap").

- *nsample* – Sets the number of samples on the MCMC chain.
- *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 iterations 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).

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

Future prediction – Here the aim is take all available data and predict what 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 investigated by adding changes to the model through ‘modification’.

Counterfactual analysis – This investigates 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 if 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 at explaining the patterns in the data.

4.4 Parallelisation

To increase computational speed BEEP can be run on multiple CPU cores. 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, 95% credible intervals, standard deviations and priors 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).
  Some algorithms provide diagnostic information to check if inference has been performed correctly, for example the effective samples size[41] (ESS), and the Gelman-Rubin statistics (GR)[42].
- **Model_specification.txt –** 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.
  - *covar* - Contains files giving information about covariates.
  - *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.
- **Simulated_data** – Under the ‘sim’ mode, this contains simulated data files corresponding to the specifications provided in the input TOML file.
- **Diagnostics** – Provides diagnostic information to inform how well the algorithm is performing:

[41] The ESS should exceed around 200 for all parameters.
[42] The GR statistic should be less than around 1.05 for all parameters.

- *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 (if used).
- *MCMC_proposals.csv* – 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 (examples of this are given in Fig. 3).

A description of the analysis can be added in the TOML by using the following command

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

and this will be displayed in the 'Model→Description' menu tab.

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.* the following specification is made in §6 EX C5:

```
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 the 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 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 various 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. In simulation mode, 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 10 build/bin/BEEP inputfile="examples/EX_#.toml" mode="map"
```

where –n sets the number of CPU cores to run on (10 in this case). Note, this uses the "map" inference methodology, but other algorithms can also be applied (*e.g.* see the file "EX_A1.toml" for different possibilities).

EX: Exemplar used in §1.4

Objective: Used in the ten-minute guide to illustrate the broad features of BEEP in a simple example.
Implementation: This model is implemented in the "example/EX.toml" file (as shown in §3.1).
Model: An SEIR model 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 this is equivalent to assuming the contact rate between individuals remains the same over time. Furthermore, a small constant external force of infection is assumed.
Data*:* Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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). '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”. The reproduction number R_t is assume to remain constant, and hence the contact rate between individuals remains the same over time. Furthermore, a small constant external force of infection is assumed.
Data: The weekly number of $S \rightarrow I$ infections and $I \rightarrow R$ 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”. This compartment has a residency which has an Erlang distribution with shape parameter 2 (equivalent to two sequentially linked 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 small 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.

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’ defines 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 small constant external force of infection.
Data: Weekly number of $S \rightarrow I$ cases and $I \rightarrow R$ 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’ defines two transition leaving the exposed E compartment (resulting in 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 infectivities. 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 small constant external force of infection.
Data: Weekly number of $E \rightarrow I$ 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 initially 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. This state is specified in the "init_pop.csv" file in the data directory, and incorporated into the TOML file through the 'init_pop' command.
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 small constant external force of infection. The population is demographically stratified by sex.
Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 recovered population).
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 small constant external force of infection. The population is demographically stratified by sex.
Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ cases, and measurements on the total infected I and population fraction of recovered individuals R, all stratified by sex. Also the total recoveries $\mathrm{I}\rightarrow\mathrm{R}$ 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 of errors 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 small constant external force of infection.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 small constant external force of infection.
Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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: Apply a Poisson observation model. This 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 at other times. 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 small constant external force of infection.
Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 the 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 small constant external force of infection.
Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 a unique spline name set 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 small constant external force of infection.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 (see Glossary).

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 small constant external force of infection.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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: Introduce a simple model which includes spatial stratification.

Implementation: 'areas' defines a file which provides information about the population, in this case four geographical regions. 'geo_mixing_matrix' defines a file which sets 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 small constant external force of infection.

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

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

EX C2: SIRD spatial model with area 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 small constant external force of infection.

Data: Weekly number of $\mathrm{S}\rightarrow\mathrm{I}$ cases and $\mathrm{I}\rightarrow\mathrm{R}$ measurements, separately for the different regions.

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

EX C3: SIRD model with demographic stratification

Objective: Provide 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 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 $\mathrm{S}\rightarrow\mathrm{I}$ cases and $\mathrm{I}\rightarrow\mathrm{D}$ 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: Incorporate age stratification into a model.

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 contacts 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 small constant external force of infection.

Data: Weekly number of S→I cases and I→R measurements separately for the different age groups.
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 later in EX E4 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 (see Appendix C). 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 small 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.

EX C6: SIRD model with different disease strains

Objective: Model which incorporated different disease strains.
Implementation: The 'strains' command is used to define the properties of discrete strains.
Model: An SIRD branching model with two strains. Individuals are assumed to have a different susceptibility to these strains. The infected individuals can either recover or die, with associated branching probabilities. A baseline reproduction number R_t remains constant but it is modified by a strain-dependent factor. A small constant external force of infection is assumed.
Data: Weekly number of S→I cases and I→D deaths, separately for the different strains.
Inference: Strain-dependent susceptibility and factor increase in reproduction number, as well as the baseline R_t, infectious period and branching probabilities.

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 branching 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 small constant external force of infection is assumed.
Data: Weekly number of S→I cases and I→D deaths.

Inference: Fixed effect for the temperature covariate, reproduction number R_t, infectious period and branching probabilities.

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 branching 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 small constant external force of infection is assumed.
Data: Weekly number of $S \rightarrow I$ cases and $I \rightarrow D$ deaths.
Inference: Fixed effect for the temperature covariate, reproduction number R_t, infectious period and branching probabilities.

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 discrete "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 branching 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 small constant external force of infection is assumed.
Data: Weekly number of $S \rightarrow I$ cases and $I \rightarrow D$ 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 small constant external force of infection.
Data: Weekly number of $S \rightarrow I$ cases and $I \rightarrow R$ 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 within and between 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. These splines are implemented using the 'age_mixing_perturb' command.

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). Time variation to this contact matrix is added (see §3.5.1). 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 small constant external force of infection is assumed.
Data: Weekly number of $\mathrm{S}\rightarrow\mathrm{I}$ cases and $\mathrm{I}\rightarrow\mathrm{D}$ 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 the external force of infection η_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. Piecewise linear splines are used to represent time variation in the reproduction number R_t and external force of infection η_t.
Data: Weekly number of $\mathrm{S}\rightarrow\mathrm{I}$ cases and $\mathrm{I}\rightarrow\mathrm{R}$ 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 (see Appendix C). 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 v
Data: Weekly number of $\mathrm{S}\rightarrow\mathrm{I}$ cases and $\mathrm{I}\rightarrow\mathrm{D}$ 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[43] (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. Beyond the vertical dashed line (labelled "Prediction start") we see 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 small constant external force of infection.

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

Prediction: Simulates system dynamics beyond available dataset.

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

[43] By default 200 posterior samples are generated, but this can be explicitly specified using 'nsample'.

dashed line (labelled “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 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 small constant external force of infection.
Data: Weekly number of $\mathrm{S}\rightarrow\mathrm{I}$ cases and $\mathrm{I}\rightarrow\mathrm{R}$ recoveries.
Prediction: Simulates system dynamics beyond available dataset assuming splines remain constant after ‘end’.

EX F3: Scenario analysis

Objective: “Scenario analysis” 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 the 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 (labelled “Prediction start”) are the future predictions from the model. The modification applied to the system is after 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 small constant external force of infection.
Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ cases and measurements on the total infected I and recovered R populations.
Scenario analysis: 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 modification 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 (labelled "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 small constant external force of infection.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 small constant external force of infection.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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 set.

EX F6: Posterior predictive checking

Objective: Posterior predictive checks are a way to verify 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 small constant external force of infection.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ 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: Introduce a model applicable to publically available data sources for the 2020 Covid-19 pandemic.

Model: A more complicated compartmental model is needed to capture Covid-19 dynamics. 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. A small constant external force of infection is assumed.

Data: Weekly number of $\mathrm{E}\rightarrow\mathrm{I}$ cases, $\mathrm{C}\rightarrow\mathrm{H}$ hospital admissions, $\mathrm{H}\rightarrow\mathrm{D}$ 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 and branching probabilities.

EX G2: Age-structured Covid-19 Model

Objective: Introduce an age-structured model applicable to 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 complex compartmental model is needed to capture Covid-19 dynamics. 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 $E \rightarrow I$ cases, $C \rightarrow H$ hospital admissions, $H \rightarrow D$ 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 and age-stratified branching probabilities.

EX G3: Age-structured Covid-19 Model with time variation in age contact

Objective: Builds on EX G2 by allowing for time variation in the relative rates of mixing between individuals in the same and different age groups, *e.g.* this is important information which can be used to assess the effect of school closures and other government interventions.

Model: The population is stratified into 10 different age groups: 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80+ and a group for care home residents. A complex compartmental model is needed to capture Covid-19 which 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 $E \rightarrow I$ cases, $C \rightarrow H$ hospital admissions, $H \rightarrow D$ 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 relative rates of mixing between different age groups, the time variation in the reproduction number R_t and age-stratified branching probabilities.

License and 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 (2022).

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 different disease strains (although they could easily be incorporated below by simply changing every instance of R_t with $R_t\psi_s$[44]).

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 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). $R_t^{\text{eff}}$is always less than R_t and if it reduces below 1, 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:

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

$$\lambda_{a,d,t} = \beta_t \sigma_d \alpha_{a,t} \sum_{a',d',c} M_{a,a',t} A_{d,d',t} i^c N^c_{a',d',t}, \tag{A37}$$

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 (A37) we see that

$$\beta_t = r_t R_t. \tag{A38}$$

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[45]. 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[46] 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

$$\frac{d\mathbf{v}}{dt} = (\mathbf{F}_t - \mathbf{\Sigma})\mathbf{v}, \tag{A39}$$

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.(A37)):

$$\mathbf{F}_t = \beta_t \left\langle \alpha_{a,t} \right\rangle_a \begin{bmatrix} 0 & 0 & \phi_M \sigma_M A_{M,M,t} i^I & \phi_M \sigma_M A_{M,F,t} i^I \\ 0 & 0 & \phi_F \sigma_F A_{F,M,t} i^I & \phi_F \sigma_F A_{F,F,t} i^I \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{A40}$$

[45] For simplicity we take the case when the Erlang distribution is a simply exponential by setting k=1.
[46] The superscript "T" strands from transpose, and this converts a row vector into a column vector.

where $\langle \ldots \rangle_a$ denotes an average of the area effects[47], 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 $\boldsymbol{\Sigma}$, representing transitions between and leaving the infected states Ω, is given by:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \frac{1}{m_M} & 0 & 0 & 0 \\ 0 & \frac{1}{m_F} & 0 & 0 \\ -\frac{1}{m_M} & 0 & \frac{1}{r_M} & 0 \\ 0 & -\frac{1}{m_F} & 0 & \frac{1}{r_F} \end{bmatrix}, \quad \text{(A41)}$$

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.(A41) is given by

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} m_M & 0 & 0 & 0 \\ 0 & m_F & 0 & 0 \\ r_M & 0 & r_M & 0 \\ 0 & r_F & 0 & r_F \end{bmatrix}. \quad \text{(A42)}$$

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.(A42) we see that if an individual enters state $\mathrm{E_M}$ (at the top) it will stay in $\mathrm{E_M}$ for time m_M, spend no time in $\mathrm{E_F}$ (individuals do not change their demographic state), time r_M in state $\mathrm{I_M}$, and no time in state $\mathrm{I_F}$.

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[48]

$$\mathbf{w}_{g+1} = \mathbf{K}_t \mathbf{w}_g \quad \text{where} \quad \mathbf{K}_t = \mathbf{F}_t \boldsymbol{\Sigma}^{-1}. \quad \text{(A43)}$$

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

[47] This average is weighed by the populations in each of the areas a.

[48] In fact because individuals usually only enter Ω through a limited number of states ($\mathrm{E_M}$ and $\mathrm{E_F}$ 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.

average Σ_{ji}^{-1} in each state j. But in doing so Eq.(A40) 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.(A43).

Next we ask the question what happens when we iterate Eq.(A43) over many generations? It turns out that this equation can be solved[49], 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.(A40) 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.(A43). The reason lies in the fact that the number of generations for $\mathbf{w}$ to actually converge on the eigenvector $\mathbf{e}$ with largest eigenvalue becomes unfeasibly 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[50] 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

$$\mathbf{F}_t' = \frac{\mathbf{F}_t}{\beta_t}. \quad \text{(A44)}$$

From Eq.(A40) 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:

[49] 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.

[50] 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.

$$\mathbf{K}'_t = \mathbf{F}'_t \boldsymbol{\Sigma}^{-1}. \tag{A45}$$

The largest eigenvalue of this matrix gives R_t divided by β_t. From Eq.(A38) 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[51].

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 (A40) is modified to now give

$$\mathbf{F}'_{sus,t} = \frac{\mathbf{F}_{sus,t}}{\beta_t} = \left\langle \alpha_{a,t} \right\rangle_a \begin{bmatrix} 0 & 0 & \phi^{sus}_{M,t}\sigma_M A_{M,M,t} i^I & \phi^{sus}_{M,t}\sigma_M A_{M,F,t} i^I \\ 0 & 0 & \phi^{sus}_{F,t}\sigma_F A_{F,M,t} i^I & \phi^{sus}_{F,t}\sigma_F A_{F,F,t} i^I \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{A46}$$

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

$$\mathbf{K}'_{sus,t} = \mathbf{F}'_{sus,t} \boldsymbol{\Sigma}^{-1}. \tag{A47}$$

If we define the largest eigenvalue of $\mathbf{K}'_t$ in Eq.(A45) to be λ_t, and for $\mathbf{K}'_{sus,t}$ in Eq.(A47) to be $\lambda_{sus,t}$, the effective reproduction number can be related to R_t through

$$R_t^{eff} = \frac{\lambda_{sus,t}}{\lambda_t} R_t. \tag{A48}$$

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

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

element in the eigenvector e_i. Equation (A42) shows a matrix Σ_{ji}^{-1}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.(A40)). The average timing of those new infections is given by the elements of the following vector:

$$\boldsymbol{\tau} = (\tfrac{1}{2}m_M, \tfrac{1}{2}m_F, m_M + \tfrac{1}{2}r_M, m_F + \tfrac{1}{2}r_A)^T. \tag{A49}$$

Combining these contributions yields the final result

$$T_{gen} = \frac{\sum_{k,j,i} F_{kj}\tau_j \Sigma_{ji}^{-1} e_i}{\sum_{k,j,i} F_{kj}\Sigma_{ji}^{-1} e_i}. \tag{A50}$$

Appendix B: Solving iterative matrix equations

This appendix outlines the solution to the matrix equation in Eq.(A43). Note, we drop the index *t* because the initial spread of the disease is assumed to be fast compared to the timescale over which model parameters change (such as the transmission rate β_t).

We refer to $\mathbf{e}$ as an "eigenvector" and λ as an "eigenvalue" of a matrix $\mathbf{K}$ if it solves the equation

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

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 $\mathbf{U}$=[$\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$,…] 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.(A43) gives

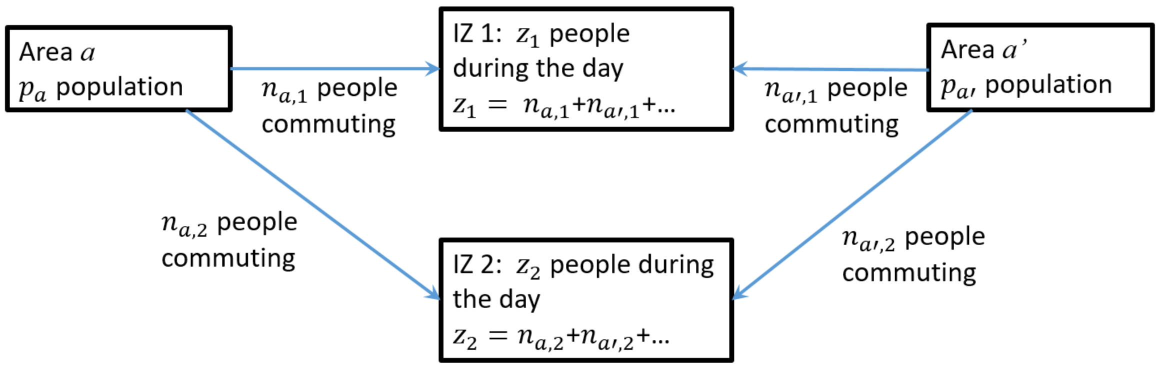
$$\begin{aligned}\mathbf{w}_g &= \mathbf{K}^{g-1}\mathbf{w}_1 \\ &= \left(\mathbf{U}\begin{bmatrix} \lambda_1 & 0 & 0 & \cdots \\ 0 & \lambda_2 & 0 & \cdots \\ 0 & 0 & \lambda_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}\mathbf{U}^{-1}\right)^{g-1}\mathbf{w}_1 \\ &= \mathbf{U}\begin{bmatrix}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

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[52] (MSOAs) in England and Wales and intermediate zones[53] (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.


Area a
p_a population
$n_{a,1}$ people commuting
IZ 1: z_1 people during the day
$z_1 = n_{a,1}+n_{a',1}+\ldots$
$n_{a',1}$ people commuting
Area a'
$p_{a'}$ population
$n_{a,2}$ people commuting
IZ 2: z_2 people during the day
$z_2 = n_{a,2}+n_{a',2}+\ldots$
$n_{a',2}$ people commuting

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

[52] These are 7,201 areas covering England and Wales.
[53] These are 1,279 areas covering Scotland.

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:

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

This matrix is symmetric, as would be expected since the overall number of contacts between individuals in different areas must be the same.

Internal to BEEP 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)).

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.* [14], POLYMOD [3] and BBC Pandemic [4]) have published estimates for a related but 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 $\mathbf{K}$ 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 $\mathbf{C}$ 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 $\mathbf{C}$ generated from **K** will not be precisely symmetric 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 by updating $\mathbf{C}$ according to

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

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:

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

  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, 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 0.01 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:

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

  where $P(x|\mu)$ is the Poisson probability distribution for x given a mean μ.
- **Negative binomial** – Similarly:

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

  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 in Eq.(17) is estimated. This is achieved by assuming that the potential stochastic paths the system can take are approximated by a multivariate normal distribution in the compartmental populations.

Before showing the derivation, in is necessary to first explain terminology. In §2.1.3 populations $p_{a,d,c,t}$ were index by area a, demographic group d and compartment c, as well as time t). Here it is convenient to amalgamate a, d and c into a single new compartmental index "c", which uniquely references individuals in a particular area, demographic state and disease status[54]. The population in compartment c at time t is denoted by p_c^t. Similarly, j indexes transitions between these new compartments.

The MVN that represents the stochastic distribution in the compartmental populations has a mean μ_c^t and covariance matrix K_{cd}^t (between compartments c and d). We now derive equations that describe the dynamic variation in $\boldsymbol{\mu}^t$ and $\mathbf{K}^t$ over time. The starting point is the tau-leaping algorithm:

$$p_c^{t+\tau} = p_c^t + \sum\nolimits_j T_{cj} n_j^t(\lambda_j^t), \tag{F1}$$

which shows how the population in p_c^t is changed at a later time point $t+\tau$. The sum j goes over all transitions within the system. n_j^t is the Poisson distributed number of transitions between t and $t+\tau$ with the mean transition number being given by λ_j^t. The matrix T_{cj} 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), or otherwise zero. For the SIR model

$$T = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}, \tag{F2}$$

where the three rows represent 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:

$$\lambda_j^t(\mathbf{p}^t) \simeq \lambda_j^t(\boldsymbol{\mu}^t) + \sum\nolimits_d \left.\frac{d\lambda_j^t}{dp_d}\right|_{\boldsymbol{\mu}^t} (p_d^t - \mu_d^t) + \ldots \tag{F3}$$

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.(F1) gives

$$p_c^{t+\Delta t} = p_c^t + \sum\nolimits_j T_{cj} n_j^t \left(\lambda_j^t(\boldsymbol{\mu}^t) + \sum\nolimits_d \left.\frac{d\lambda_j^t}{dp_d}\right|_{\boldsymbol{\mu}^t} (p_d^t - \mu_d^t) \right). \tag{F4}$$

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

[54] Consequently the number of new compartments = number of areas × the number of demographic classifications × the number of disease status classifications.

$$\begin{aligned}\mu_c^{t+\tau} &= \left\langle p_c^{t+\tau} \right\rangle \\ &= \mu_c^t + \sum\nolimits_j T_{cj} \lambda_j^t(\boldsymbol{\mu}^t).\end{aligned} \tag{F5}$$

The mean transition number λ_j^t can alternatively be represented by the transition rate r_j^t multiplied by the timestep τ. Consequently Eq.(F5) can be re-expressed (for the limit of small τ) as

$$\frac{d\mu_c^t}{dt} = \sum\nolimits_j T_{cj} r_j^t(\boldsymbol{\mu}^t). \tag{F6}$$

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

Subtracting Eq.(F5) from Eq.(F4) gives

$$p_c^{t+\tau} - \mu_c^{t+\tau} = p_c^t - \mu_c^t + \sum\nolimits_j T_{cj} \left[n_j^t \left(\lambda_j^t(\boldsymbol{\mu}^t) + \sum\nolimits_m \left. \frac{d\lambda_j^t}{dp_m} \right|_{\boldsymbol{\mu}^t} (p_m^t - \mu_m^t) \right) - \lambda_j^t(\boldsymbol{\mu}^t) \right]. \tag{F7}$$

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

$$\begin{aligned} K_{cd}^{t+\tau} &= \left\langle (p_c^{t+\tau} - \mu_c^{t+\tau})(p_d^{t+\tau} - \mu_d^{t+\tau}) \right\rangle \\ &= \left\langle \left(p_c^t - \mu_c^t + \sum\nolimits_j T_{cj} \left[n_j^t \left(\lambda_j^t(\boldsymbol{\mu}^t) + \sum\nolimits_m \left. \frac{d\lambda_j^t}{dp_m} \right|_{\boldsymbol{\mu}^t} (p_m^t - \mu_m^t) \right) - \lambda_j^t(\boldsymbol{\mu}^t) \right] \right) \right. \\ &\qquad \left. \left(p_d^t - \mu_d^t + \sum\nolimits_k T_{dk} \left[n_k^t \left(\lambda_k^t(\boldsymbol{\mu}^t) + \sum\nolimits_n \left. \frac{d\lambda_k^t}{dp_n} \right|_{\boldsymbol{\mu}^t} (p_n^t - \mu_n^t) \right) - \lambda_k^t(\boldsymbol{\mu}^t) \right] \right) \right\rangle \end{aligned} \tag{F8}$$

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

$$K_{cd}^{t+\tau} = K_{cd}^t + \sum\nolimits_j T_{cj} T_{dj} \lambda_j^t(\boldsymbol{\mu}^t) + \left(\sum\nolimits_{j,m} T_{cj} \left. \frac{d\lambda_j^t}{dp_m} \right|_{\boldsymbol{\mu}^t} K_{dm}^t \right) + \left(\sum\nolimits_{k,n} T_{dk} \left. \frac{d\lambda_k^t}{dp_n} \right|_{\boldsymbol{\mu}^t} K_{cn}^t \right). \tag{F9}$$

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

$$\frac{dK_{cd}^t}{dt} = \sum\nolimits_j T_{cj} T_{dj} \lambda_j^t(\boldsymbol{\mu}^t) + \sum\nolimits_{j,m} \left(T_{cj} K_{dm}^t + T_{dj} K_{cm}^t \right) \left. \frac{d\lambda_j^t}{dp_m} \right|_{\boldsymbol{\mu}^t}. \tag{F10}$$

The dynamic Eqs. (F6) and (F10) can be integrated to estimate how the population MVN changes over time. Next we look at what happens to $\boldsymbol{\mu}^t$ and $\mathbf{K}^t$ as a result of observations made on the system.

Observations

Observations on the system restrict the possible states the posterior state can take and they come in two types:

Population measurements – Suppose i indexes population observations and Y_i estimates the population in compartment(s) C_i (which may be one or a group of compartments) at time T_i with uncertainty σ_i. The observation model probability for measurement i is given by

$$\pi(y_i \,|\, \mathbf{p}^{T_i}) = \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\frac{1}{2\sigma_i^2}\left(\left(\sum_{c\in C_i} p_c^{T_i}\right)-Y_i\right)^2}, \tag{F11}$$

which is a univariate normal distribution centred at the observation with standard deviations σ_i. In the case of population fraction measurements, Y_i is simply replaced $Y_i \times N$, where N is the total population size.

Transition measurements – Suppose i indexes observations and Y_i estimates the number of transitions down a given transition j over a time period from T_i^1 to T_i^2. This could either be a short time interval (*e.g.* just a daily measurement in a time-series) or a long interval (as would be used for a marginal observation). To proceed, each observation i needs to be converted into an effective population measurement. This can be explained by means of a simple example. Consider the infection S→E transition in the SEIR model in Fig. 1. One way to measure the number of these transitions would be to just count how many happen between T_i^1 to T_i^2. An alternative is to calculate the difference in the entire population in the E, I and R compartments *between* T_i^1 and T_i^2. Such an approach works because any individual which passes down S→E must inevitably remain in either E, I and R. The observation model probability becomes

$$\pi(y_i \,|\, \mathbf{p}^{T_i^2}) = \frac{1}{\sqrt{2\pi\sigma_i}} e^{-\frac{1}{2\sigma_i^2}\left(\left(\sum_{c\in C_i} p_c^{T_i^2}\right)-\left(Y_i+\sum_{c\in C_i} \mu_c^{T_i^1}\right)\right)^2}, \tag{F12}$$

where C_i includes the final compartment of the transition and all other compartments connected to it. This approach relies on two restrictions: 1) backward transitions in the model are not allowed[55] and 2) merging of compartments is not allow. The latter is unrestrictive, because any model with merging can we "uncoiled" by duplication of compartments such that no such merging occurs (this is actually what BEEP automatically does in the background).

Combining simultaneous observations

Often several observations are made concurrently at the same time point (for example we might know both the number in hospital and the number recovered at a series of time points). Therefore

[55] So, for example, it would not be possible include waning immunity.

here we order observations into time “slices” indexed by *s* at times t_s. For each slice the observation model is given by

$$\pi(y_s \,|\, \mathbf{p}^{t_s}) = \prod\nolimits_{i\in s} \pi(y_i \,|\, \mathbf{p}^{t_s}), \tag{F13}$$

where the produce *i* goes over all observations in the slice. Due to the fact that the distributions in Eqs. (F11) and (F12) are Gaussian, the overall probability distribution for a slice can be written as proportional to a multivariate normal distribution

$$\pi(y_s \,|\, \mathbf{p}^{t_s}) = \omega_s f_{MVN}(\mathbf{p}^{t_s} \,|\, \mathbf{m}_s, \mathbf{\Omega}_s) \tag{F14}$$

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

Calculating the likelihood

Usually the initial state for the system $\boldsymbol{\mu}^0$ 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 (F6) and (F10) 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 current estimated distribution in states and the observation model for the time splice (see note below regarding removal of the susceptible states) :

$$f_{MVN}(\mathbf{p}^{t_s} \,|\, \boldsymbol{\mu}^{t_s}, \mathbf{K}^{t_s}) \times \omega_s f_{MVN}(\mathbf{p}^{t_s} \,|\, \mathbf{m}_s, \mathbf{\Omega}_s). \tag{F15}$$

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

$$L_s f_{MVN}(\mathbf{p}^{t_s} \,|\, \mathbf{m}_s, \mathbf{\Omega}_s), \tag{F16}$$

where

$$\begin{aligned} \left(\mathbf{K}^{t_s}_{new}\right)^{-1} &= \left(\mathbf{K}^{t_s}\right)^{-1} + \left(\Omega_s\right)^{-1}, \\ \left(\mathbf{K}^{t_s}_{new}\right)^{-1} \boldsymbol{\mu}^{t_s}_{new} &= \left(\mathbf{K}^{t_s}\right)^{-1} \boldsymbol{\mu}^{t_s} + \left(\Omega_s\right)^{-1} \mathbf{m}_s, \end{aligned} \tag{F17}$$

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

$$L_s = \omega_s \sqrt{\frac{\left|\mathbf{K}^{t_s}_{new}\right|}{(2\pi)^d \left|\mathbf{K}^{t_s}\right| \left|\Omega_s\right|}} e^{-\frac{1}{2}\left(\boldsymbol{\mu}^{t_s T}\left(\mathbf{K}^{t_s}\right)^{-1}\boldsymbol{\mu}^{t_s} + \mathbf{m}_s^T\left(\mathbf{\Omega}^{t_s}\right)^{-1}\mathbf{m}_s - \boldsymbol{\mu}^{t_s T}_{new}\left(\mathbf{K}^{t_s}_{new}\right)^{-1}\boldsymbol{\mu}^{t_s}_{new}\right)}. \tag{F18}$$

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

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

$$L(y\,|\,\theta) = \prod\nolimits_s L_s. \tag{F19}$$

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 - Removal of susceptible states: The MVN product in Eq.(F15) is, in reality, improper due to conservations which underlie the system dynamics (in particular, compartmental models preserve the overall number of individuals $N_{a,d}$ in area a and demographic category d). This can be overcome by first removing the susceptible S states from the matrices in Eq.(F15), performing the MVN multiplication, and then reconstruct the S states once again[56]).

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:

Maximum *a posteriori* estimate (MAP)

First CMA-ES is used to estimate the parameter set θ_{MAP} with the maximum *a posteriori* probability. The Hessian matrix $\mathbf{H}$ is then numerically calculated (using a simple finite-difference scheme):

$$H_{i,l} = \frac{\partial^2}{\partial\theta_i \partial\theta_j} \log\left(L(y\,|\,\theta)\right). \tag{G1}$$

The model evidence is approximated using Laplace's method as

$$\pi(y) \simeq \frac{(2\pi)^{k/2}}{\sqrt{|\mathrm{H}|}} L\left(y\,|\,\theta_{MAP}\right)\pi\left(\theta_{MAP}\right), \tag{G2}$$

where k is the number of model parameters.

Metropolis coupled MCMC (MC^3)

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

$$\phi_k = \left(\tfrac{K-k}{K-1}\right)^4 \phi_{post}. \tag{G3}$$

The model evidence is calculated using

[56] This is possible because the susceptible compartments $p_{a,d,S,t}$ can be calculated directly from others by $N_{a,d} - \sum_{c \neq S} p_{a,d,c,t}$.

$$\pi(y \mid \phi_1) = \prod_{k=2}^{K} \left(\tfrac{1}{N} \sum_{i=1}^{N} \pi(y \mid \xi_i^k)^{\phi_{k-1}-\phi_k} \right), \tag{G4}$$

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

Particle annealed sampling using MBPs (PAS-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

$$\pi(y \mid \phi_G) = \prod_{g=1}^{G-1} \left(\tfrac{1}{N} \sum_{i=1}^{N} \pi(y \mid \xi_i^g)^{\phi_{g+1}-\phi_g} \right), \tag{G5}$$

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:

$$\theta_{new} \sim MVN(\theta_i^g, \Sigma). \tag{G6}$$

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

$$w_{new} = \frac{\pi(\theta_{new})}{\sum_{i=1}^{N} w_i^g \, p_{MVN}(\theta_{new} \mid \theta_i^g, \Sigma)}, \tag{G7}$$

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[57] is given by:

$$\pi(y \mid EF_{cutoff}) = \frac{\sum_\text{new accepted} w_{new}}{\sum_\text{new} w_{new}}, \tag{G8}$$

[57] 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

$$\pi(y \mid EF_{\text{cutoff}}^{G}) = \prod_{g=1}^{G-1} \left[\frac{\sum_{g'=1}^{g} \sum_{i=1}^{N} H\left(EF_{\text{cutoff}}^{g'+1} - EF(\xi_i^{g'}) \right)}{\sum_{g'=1}^{g} \sum_{i=1}^{N} H\left(EF_{\text{cutoff}}^{g'} - EF(\xi_i^{g'}) \right)} \right], \qquad \text{(G9)}$$

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 one (or a few) of the columns and rows of the basic age 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.
age_mixing_ perturb **§3.5.1**	This is used to modify all the rows and columns of the basic age mixing matrix as a function of time (here the file "amp.csv" informs a spline, see §3.3). *E.g.* age_mixing_perturb = [{ agecat='age0-19', bp='[bp:amp.csv]', param='[Par0-19:amp.csv]', value='[Value0-19:amp.csv]', prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'}, { agecat='age20-39', bp='[bp:amp.csv]', param='[Par20-39:amp.csv]', value='[Value20-39:amp.csv]', prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'}, { agecat='age40-59', bp='[bp:amp.csv]', param='[Par40-59:amp.csv]', value='[Value40-59:amp.csv]', prior='MDir(0.3)', smooth='0.1', smooth_type='log_smooth'}, { agecat='age60+', bp='[bp:amp.csv]', param='[Par60+:amp.csv]', value='[Value60+:amp.csv]', prior='MDir(0.3)', smooth='0.1',

	smooth_type='log_smooth'}] *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 longitude 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)[58] 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).

[58] 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.

	inf_X – Sets a model parameter that specifies the relative infectivity for that compartment (if not set it is assumed to be zero).
cor_max **§4.2.2**	Sets the maximum correlation in parameters between one generation and the next (by default set to 0.5). If this value is set higher, generations will proceed quicker but particles will exhibit a higher degree of correlation. E.g. cor_max = 0.9
cor_max_last **§4.2.2**	Sets the maximum correlation in parameters between the penultimate and last generation (by default set to 0.5). Typically if ‘cor_max’ is set to a high value then ‘cor_max_last’ will be set lower to ensure that particles for the final posterior plot are relatively uncorrelated. E.g. cor_max_last = 0.5 • *cpu_time* – Iterates generations until a specified CPU time is reached (in minutes), at which point the algorithm terminates and outputs are plotted.
cpu_time **§4.2.1, §4.2.2, §4.2.3, §4.2.5**	Iterates generations until a specified CPU time is reached (in minutes), at which point the algorithm terminates and outputs are plotted. E.g. cpu_time = 10
cutoff **§4.2.4**	Specifies the cut-off in the error function (‘abc’ mode). *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). Examples of filtering on age or strain are given by ‘democats_filt="age:0-9"’ and ‘democats_dep="strain:s1"’, respectively.

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 or strain dependency ‘democats_dep="age"’ or ‘democats_dep="strain"’ are used, respectively. *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 'pas' 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.(19) (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). "map" – Performs inference using maximum *a posteriori* (MAP) estimation. "abcmbp" – Performs inference using the ABC-MBP algorithm. "pas" – Performs inference using the PAS-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 **§4.2.6-8**	Sets the number of ‘burn-in’ steps when performing MCMC (used in the ‘pmcmc’, ‘mcmcmbp’ and ‘mc3’ modes). *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 **§4.2.1, §4.2.2, §4.2.3, §4.2.5**	Sets the number of generations (options in the ‘map’, ‘abcmbp’, ‘abcsmc’ and ‘pas’ 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 **§4.2.1, §4.2.2, §4.2.3, §4.2.6**	The number of particles (optional in 'map' and required in the 'abcmbp', 'pas' 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 **§4.2.1, §4.2.4-8**	Sets the number of samples to be generated (used in the 'map', '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 'pas' 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	Defines linear splines that are used as factors to relate the 'observation' quantity in 'data_tables' with the actual data observed.

§3.7.3	*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).
posterior_particle **§4.2.1**	In 'map' mode, this sets the number of particles used in the sequential Monte-Carlo scheme that generates the final posterior state samples (optional, by default set to 20). Making this number larger improves the accuracy with which state samples are generated. *E.g.* posterior_particle = 100
plot_param_values **§3.11.5**	Places simulated values onto output plots (optional, by default set to false). *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.
quench_factor **§4.2.3**	Sets the rate of quenching in the ‘pas’ 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
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** **§3.9**	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 approximates the continuous time Gillespie algorithm, but take computationally longer to run. *E.g.* steps_per_unit_time = 4
strains **§3.8**	Used to specify different strains of virus. *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 **§3.7.2**	When data contains thresholded values, this sets the string used in the data to represent this fact (optional, by default set to "*"). *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 **§3.1**	This determines the format in which times are represented in the input TOML file and any data files. *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 **§3.11.4**	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. • *E.g.* time_labels = [{name="Lockdown", time="2020-03-23"}]
trans **§3.4.1**	Defines transitions between compartments. *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] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evolutionary computation,* vol. 11, no. 1, pp. 1-18, 2003.

[6] N. Friel, M. Hurn, and J. Wyse, "Improving power posterior estimation of statistical evidence," *Statistics and Computing,* vol. 24, no. 5, pp. 709-723, 2014.

[7] C. Pooley, S. Bishop, and G. Marion, "Using model-based proposals for fast parameter inference on discrete state space, continuous-time Markov processes," *Journal of The Royal Society Interface,* vol. 12, no. 107, p. 20150225, 2015.

[8] C. Pooley, S. Bishop, A. Doeschl-Wilson, and G. Marion, "Posterior-based proposals for speeding up Markov chain Monte Carlo," *Royal Society open science,* vol. 6, no. 11, p. 190619, 2019.

[9] W. R. Gilks and G. O. Roberts, "Strategies for improving MCMC," *Markov chain Monte Carlo in practice,* vol. 6, pp. 89-114, 1996.

[10] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* vol. 72, no. 3, pp. 269-342, 2010.

[11] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate Bayesian computation in population genetics," *Genetics,* vol. 162, no. 4, pp. 2025-2035, 2002.

[12] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. Stumpf, "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," *Journal of the Royal Society Interface,* vol. 6, no. 31, pp. 187-202, 2009.

[13] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the american statistical association,* vol. 90, no. 430, pp. 773-795, 1995.

[14] 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.