BEEPmbp

BEEP-MBP V1.0

Bayesian Estimation of Epidemiological Parameters with Model-based Proposals

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

Table of Contents

THIS DOCUMENT IS CURRENTLY BEING WRITTEN, SO WE APPOLOGISE FOR ANY INACCURACIES.

1 Introduction

Compartmental models have long been used as a means of understanding the collective dynamics of interacting agents, with notable applications in epidemiology and ecology. A simple example is shown in Fig. 1. Here individuals start in the susceptible compartment, and when infected they make transitions to the exposed, infectious and recovered states.

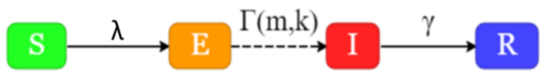

Fig 1. Compartmental model. This describes how an individual's infection status changes over time. S: susceptible, E: exposed, I: infectious, R: recovered. Individuals are infected at a rate given by the force of infection λ, undergo a non-infectious incubation period given by gamma distribution Γ, and finally recover at a rate γ.

Numerous refinements to this basic model can be made: 1) inclusion of various degrees of infection severity (*e.g.* asymptomatic states, all the way up to hospitalised and dead), 2) age structure (*e.g.* to capture age differences in susceptibility or allow for different age groups in the population to interact with each other to a greater or lesser extent), and 3) spatial variation (to incorporate the fact that an infected individual is more likely to transmit their infection to another person locally than elsewhere).

1.1 This software

BEEPmbp (Bayesian Estimation of Epidemic Parameters using Model-Based Proposals) is a general-purpose software tool for simulating and performing inference on compartmental models. Inference is the method by which suitable model parameters are chosen from available data. To perform inference BEEPmbp accepts a variety of population-based data (time series giving the rate of transitions, populations in different compartments and marginal distributions). For example, when analysing COVID-19 disease transmission the following data are used: daily hospitalisations, deaths, populations in hospital as well as overall age distributions for these quantities.

In BEEPmbp priors on model parameters can be specified from a large range of possibilities. The outputs consist of posterior estimates, a pdf containing numerous plots relating to posterior variation in the model parameters and system state as well as diagnostic information.

BEEPmbp features:

- Specify an arbitrary compartmental model.
- Incorporate spatial and age-structured models.
- Time-varying splines to capture variation in reproduction number R(t) and the external force of infection.
- Split population into arbitrary demographic classifications (age, sex).
- Incorporate susceptibility variation for different demographic groups.
- Add area-based covariates to modify the force of infection (*e.g.* population density).

- Incorporate a user specified age-mixing matrix (along with potential time modification).
- Specify matrix for mixing between different areas (along with potential time modification).
- Perform posterior predictive checks as well as analyse counterfactuals.
- Choose from 6 different inference algorithms.
- Efficient parallel implementation.

1.2 Getting started

The code should be downloaded from git using the "--recursive" flag to ensure that submodules are included. For example:

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

If you have downloaded without the --recursive flag, you can update the submodules with

```
git submodule update –init
```

You need to have MPI installed to compile and run this code. The code uses mpicxx to compile, so this must be available. On the command line navigate to the BEEPmbp directory and type

```
make
```

to compile the software.

1.3 A five-minute guide

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

- Type '`./beepmbp inputfile="examples/SEIR.toml" mode="sim"`'. This simulates from the model in Fig. 1, with the software outputs placed into the directory “Output_SEIR”.
- In this output directory, view the file “Graphs.pdf”. This gives various plots that summarise the system dynamics.
- In the sub-directory “Simulate_data” observe a file which represents simulated data for the daily number of infection transitions in the system. We now look to use inference from this data to generate an estimate of the model parameters.
- Type ‘mpirun -n 10 ./beepmbp inputfile="examples/SEIR.toml" mode="abcmbp" nparticle=200 ngeneration=10’ to run inference.
- If we now view the file “Graphs.pdf” we see that model parameter and the system state are estimated based on the data.
- Looking at the file “Parameter_estimates.txt” we see posterior mean estimates for the model parameters along with 95% credible intervals.

2 Epidemiological model

Here will be a description of the model, in particular how the force of infection is formulated and how the error function is calculated…

3 The input TOML file

The input TOML file contains the information BEEPmbp needs to define the compartmental model and perform simulation or inference. A complete reference for all the commands used by BEEPmbp is given in Appendix A. The software also contains numerous examples, as discussed in section 6, that help to illustrate uses for each of these commands.

Here we go through the following very simple example (taken from the file "examples/SEIR.toml"):

```
timeformat = "year-month-day"
start = "2020-01-01"
end = "2020-07-01"
datadir = "examples/Data_SEIR"
outputdir = "Output_SEIR"
comps = [{name="S"},{name="E"},{name="I", inf="infI", inf_value="1"},{name="R"}]
trans = [{from="S", to="E", dist="Infection"},
         {from="E", to="I", dist="Erlang", k=2, mean="tE", mean_value="4"},
         {from="I", to="R", dist="Exp", mean="tI", mean_value="4"}]
R_spline = [{ param="R", value="2.0", prior="Uniform(0.4,4)"}]
efoi_spline = [{param="eta", value="1"}]
areas = "areadata.csv"
data_tables = [{type="transition", observation="E->I", file="EI.csv"}]
```

'timeformat' determines how time is represented (in both the TOML file and also any data files).

'start' and 'end' give the start and end date for the analysis.

'datadir' defines the location of the data directory, and 'outputdir' indicates where outputs from BEEPmbp are placed.

'comps' defines the compartments in the model, which are here specified to be: susceptible "S", exposed "E", infectious "I" and recovered "R" (see Fig. 1). Here 'inf' defines a parameter giving the infectiousness of the compartment (and if omitted the infectiousness is assume to be zero), with 'inf_value' setting its value to 1 (note, if the model contains multiple infectious states, setting different values can be used to determine the relative infectivity of the different states).

'trans' defines the transitions in the model (corresponding to the arrows in Fig 1). Within this 'dist' defines the distribution in time for the transition to occur. This can take three possible values: "Infection" means that transition refers to an infection (the rate for this is automatically calculated based on the reproduction number specified later), "Exp" denotes the exponential distribution (and a parameter giving its mean is specified), and "Erlang" denotes the Erlang distribution (with a parameter specifying mean along with a shape 'k').

'R_spline' defines how the reproduction number changes over time. If it is set to a specific parameter (as here), it will be assumed constant. However, time variation in R can easily be incorporated (*e.g.* to account for the effect of social distancing and/or government lockdown policies), as shown in section 6.1. Note here that a prior is set to a uniform distribution between 0.4 and 4. Under inference this specifies the bounds over which that parameter is expected to lie.

‘efoi_spline’ defines a spline giving the external force of infection. This is used to generate infections in the system that initiate epidemics in the first place. By default, its value is expressed as the probability of infection per unit time per 100,000 individuals (although this can be changed using the ‘efoi_factor’ command).

‘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 the example “areadata.csv” looks like this:

```
area,population
Scotland,5454000
```

One column in this file must have the heading “area”, and this refers the geographical area under consideration. This particular example gives a non-spatial analysis (focusing on a nationwide analysis of Scotland), but a list of different areas can be provided (*e.g.* see section 6.3 where local authorities are looked at). Another column must have the heading “population” (or alternatively several columns stratifying the population into different demographic groups, see section 6.5).

Finally, ‘data_tables’ defines the names and types of different data files (which can again be in “.csv” or “.txt” formats). In the example above, ‘data_tables’ reads in the number of transitions between E and I per day from the file “EI.csv” (note, when run in inference mode these files should be placed in the data directory, whereas in simulation mode these data files are actually generated in the “Simulate_data” sub-directory in the output).

3.1 Priors

In the example above all parameters were set to a particular value except for R, which was taken to have a uniform prior. In fact any specified parameter can have a prior distribution associated with it. Priors can take the following values: “Fixed(.)” to fix the parameter value, “Uniform(.,.)” for a uniform distribution, “Exp(.)” for an exponential distribution and “Gamma(.,.)” for a gamma distribution.

It is important to note that data often provides little information about some model parameters, leading to a posterior distribution almost the same as the prior. This can lead to a significant computational time increase when performing inference, therefore wherever possible it is advisable to place as restrictive priors on model parameters as possible.

3.2 Data tables

Here we provide more information about the command ‘data_tables’. This can accept three types of data:

- **Time series transition data**, *e.g.* type="timeseries", observation="E->I" gives the number of transitions per unit time between the two specified compartments. The units property (*e.g.* units=”weeks”) can be specified if data collection occurs on a different time-scale.
- **Time series population data**, *e.g*. type="population", observation="I" gives the population in the infectious compartment.
- **Marginal data,** *e.g*. type="marginal", observation="S->E". This gives the total number of transitions of the entire analysis time (or alternatively ‘start’ and ‘end’ times can be set).

Furthermore, if the data is stratified by geographical area or demographic group, this can also be incorporated (see Appendix A for details).

4 Running BEEPmbp

Once the input TOML file is specified and BEEPmbp is compiled (using the “make” command) it can be run by executing “./beepmbp” on the command line followed by a specification of the input file name. Following that, additional commands can be specified (note, these will override any existing specifications in the TOML file itself) and these are used to quickly change the analysis without having to edit the input file each time.

4.1 Simulation-based approaches

Four 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.

```
./beepmbp inputfile="file.toml" mode="sim"
```

4.1.2 Multiple simulation

This simulates from the model multiple times and outputs the distributions in states generated (see “Graphs.pdf”). This make it possible to visualise the level of inherent stochastic variation in the model itself.

```
./beepmbp inputfile="file.toml" mode="multisim" nsimulation=100
```

Here ‘nsimulation’ specifies the number of simulations to be performed.

4.1.3 Posterior predictive check

When inference is performed a series of posterior samples for the model parameters and system state are generated and saved in the “Posterior/samples” directory. Running a posterior predictive check loads up these samples and simulates from them subsequent to a specified time point t_p. This is a useful way to check that the model is reliably predicting the true data values subsequent to t_p.

```
./beepmbp inputfile="file.toml" mode="ppc" nsample=200 ppc_start="2020-4-01"
```

‘nsample’ gives the number of posterior samples to load (this must agree with the number used when inference was performed) and ‘ppc_start’ gives the time t_p (if omitted this will be set to the ‘start’ time).

4.1.4 Counterfactuals

As with posterior predictive checks, posterior samples are first loaded up. This time, however, when simulations are performed the model is altered in some way to see what effect this would have had on system dynamics.

```
./beepmbp inputfile="file.toml" mode="counter"  nsample=200
```

The model alteration is specified in the input TOML file using the ‘counterfactual’ command, *e.g.*

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

The states that after the 1st of April there is a 50% reduction in the infection transition rate comparted to what was inferred.

4.2 Inference

BEEPmbp incorporates six inference algorithms that all aim to perform essentially the same task: to generate posterior samples for model parameters and the system state. Which of these algorithms to choose, however, may depend on the scenario under consideration, with some performing computationally better or worst depending on the situation. They are split into two broad classes: those which aim to reduces an error function (ABC-MBP, ABC, ABC-SMC) and those which aim to incorporate an a specified observation model (PAIS-MBP, PMCMC , MC^3). We have found that in most cases ABC-MBP works the best for the former and PAIS-MBP for the latter.

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

This starts with a number of randomly sampled "particles" from the prior (a particle comprises of a parameter set and a system state). The algorithm proceeds through a series of generations and within each generation: a) the EF cut-off is reduced such that half the particles are copied and half are culled and b) a series of MCMC updates allow for particles to explore parameter and state space (to avoid degeneracy).

```
./beepmbp inputfile="file.toml" mode="abcmbp"  nparticle=200 ngeneration=40
```

'nparticle' sets the number of particles (200 is typical as this yields approximately 200 random samples from the posterior) and 'ngeneration' gives the number of generations (a suitable value is problem dependent, but typically this lies in the range 10-100 depending on the model complexity).

4.2.2 Particle Annealed Importance Sampling with model-based proposals (PAIS-MBP)

Whereas ABC-MBP aims to generate the posterior by reducing the error function below a cut-off value, here we assume a full observation model and aim to generate the posterior from that. PAIS-MBP starts with a number of randomly sampled "particles" from the prior. The algorithm proceeds through a series of generations and within each generation: a) the inverse temperature is increased such that on average half the particles are copied and half are culled, and b) a series of MCMC updates allow for particles to explore parameter and state space (to avoid degeneracy).

```
./beepmbp inputfile="file.toml" mode="pais"  nparticle=200 ngeneration=40
```

'nparticle' sets the number of particles and 'ngeneration' is the number of generations. Instead of specifying 'ngeneration', 'invT_final' can be set, and here the generations are looped over until the specified final inverse temperature is reached.

4.2.3 Approximate Bayesian Computation (ABC)

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

```
./beepmbp inputfile="file.toml" mode="abc"  nsample=200 cutoff_frac=0.01
```

This generates 200 posterior samples by simulating from the model 200/0.01=20000 times and choosing the 200 samples with the lowest EF. Instead of specifying 'cutoff_frac', 'cutoff' can be set and only those simulated states with EF below this cut-off are accepted (this approach has the advantage of using much less memory, because it is not necessary to store all the generated states).

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

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

‘nsample’ gives the number of samples in each generation, ‘ngeneration’ gives the number of generations and ‘cutoff_frac’ specifies how the cut-off in EF is chosen such that a certain fraction of particles are accepted for the next generation.

4.2.5 Particle Markov chain Monte Carlo (PMCMC)
This runs an MCMC chain in parameter space with a particle filter used to generate an unbiased estimate for the agreement between the model and the data.

```
./beepmbp inputfile="file.toml" mode="pmcmc"  nsample=5000 nparticle=200
```

‘nsample’ gives the number of samples on the MCMC chain (note this is typically much higher than other methods because successive samples are highly correlated) and ‘nparticle’ gives the number of particles used in the filter (a higher number here allows for a better agreement with the data but is computationally slower).

4.2.6 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.

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

‘nsample’ gives the number of samples on chain, ‘nchain’ gives the number of MCMC chains used (this needs to be sufficiently large to allow for adjacent chains to “overlap”), and ‘invT_final’ specifies the inverse temperature of the posterior.

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

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

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

5 Outputs
We give a brief description below of the various files and folders contained within the output directory once analysis is complete:

- **Parameter_estimates.txt** – If inference is performed, this file gives the posterior means and 95% credible intervals for each of the model parameters.
- **Graphs.pdf** – This visualises outputs from BEEPmbp.
- **Graphs_description.txt** – Provides information about the source data for the graphs.
- **Model_specification.txt –** This gives a summary of the model used to perform the analysis.
- **Posterior** – This folder contains files giving posterior probability distributions for the model parameters. These are arranged in a number of different ways:
  - *parameter* - Contains files for the model parameters.
  - *state* - Contains files which compare the system state with that observed in the actual data files.
  - *spline* - Contains files giving time variation in splines used within the model.
  - *susceptibility* - Contains files giving the variation in susceptibility for different demographic classes within the model.
  - *sample* - Contains files giving raw posterior samples.
  - *Rmap.txt* - For spatial models this gives the variation in R across different regions.
- **Simulated_data** – This gives simulated data files corresponding to the specifications provided in the input TOML file.
- **Diagnostics** – This provides diagnostic information to inform how well the algorithm is performing:
  - *Generation.txt* - Shows how model parameters move from the prior distribution to an approximation of the posterior distribution as a function of the generation number ('generations' are used within the ABC-MBP and ABC-SMC procedures to filter particles such that they provide a better and better approximation to the posterior).
  - *MCMC_proposals.txt* - This shows the performance of any MCMC proposals used.

6 Examples

This section goes through a series of examples that help to illustrate the power of BEEPmbp (the files for these can be found in the "examples" directory).

6.1 Incorporating time variation into the reproduction number

6.2 Adding braches into compartmental model

6.3 Adding spatial variation

6.4 Adding demographic structure

6.5 Different types of data

6.6 Modification to spatial interaction

6.7 Modification to age mixing

6.8 More than one infection transition

6.9 Variation in susceptibility

License and warranty

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

Citing BICI

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

Pooley CM, Doeschl-Wilson AB, Marion G., *BEEPmbp – A flexible tool for simulation and inference from user-defined compartmental models*. To be submitted (2021).

Acknowledgments

We would like to acknowledge the contribution of toml11 (github.com/ToruNiina/toml11), which is the C++ TOML parser that BEEPmbp uses to read the input TOML file.

References

[1] C. M. Pooley, S. C. Bishop, A. B. Doeschl-Wilson, and G. Marion, "Estimating genetic and non-genetic effects for host susceptibility, infectivity and recoverability using temporal epidemic data," *bioRxiv,* p. 618363, 2019.

[2] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics,* vol. 33, no. 3, pp. 1065-1076, 1962.

[3] R. E. Kass and A. E. Raftery, "Bayes factors," *Journal of the American Statistical Association,* vol. 90, no. 430, pp. 773-795, 1995.

[4] H. Jeffreys, *The theory of probability*. OUP Oxford, 1998.

[5] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical science,* vol. 7, no. 4, pp. 457-472, 1992.

[6] B. Carpenter *et al.*, "Stan: A probabilistic programming language," *Journal of statistical software,* vol. 76, no. 1, 2017.

Appendix A

This provides an alphabetical list of all the commands that can be used in the input TOML file:

General	
age_mixing_ matrix	Gives the name for a file containing the age-stratified contact matrix. This square matrix must have the same dimensions as the number of age groups specified in ‘ages’ (if ages has only one classification then age_mixing_matrix does not need to be specified). *E.g.* age_mixing_matrix = "BBC-Pandemic.txt"
age_mixing_ modify	This is used to modify the basic mixing matrix as a function of time. *E.g.* age_mixing_modify = [{type = "row_column", ages="20-69", param = "fac_.", bp="bp|amm.txt", value="value|amm.txt ", prior="Uniform(0,5)", smooth="0.3"}, *type* – Determines the type of the modification. • row_column – This multiplies the ages specified by subsequently defined spline.
ages	The age groups used in the analysis. *E.g.* ages = [{range="0-19", sus="sus_young", sus_value="0.7"}, {range="20-69", sus="sus_adult", sus_value="1.0"}, {range="70+", sus="sus_old", sus_value="2.0"}] *range* – The age range of individuals in a given group. *sus* – Defines a parameter giving the relative susceptibility. *sus_value* – The value for that parameter (under simulation).
areas	Gives 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.txt" The table contains the following columns: “area”, other coarser geographical scales if they are used in the data (*e.g.* Scotland), columns for covariates in the model (*e.g.* density of population), either “population” or columns for populations in different age groups, columns for other demographic categories (giving percentages).
comps	Defines the compartments in the model. *E.g.* comp = [{name="S"}, {name="I", inf="infI", inf_value="1"}, {name="R"}] *name* – Gives the name of a compartment. *inf* – Sets a model parameter that gives the relative infectivity for that compartment (if not set it is assumed to be zero). *inf_value* – Gives the value for that parameter under simulation (optional). *inf_prior* – Gives the prior specification for the parameter (optional, if not set the fixed *inf_value* is used).

	inf_trans – If there are more than one infection transitions, this gives the transition on which the infectivity acts (optional).
covars	Covariates for area. These are quantities potentially related to disease transmission and we wish to estimate the size of the effect. *E.g.* covars= [{name="density", param="den", value="0.1", func="log"}] *name* – The name of the covariate (note this heading must appear in the ‘areas’ file). *param, value, prior* – Defines the parameter to be estimated. *func* - Gives the functional transformation which can be 'linear' or 'log'.
counterfactual	In mode ‘counter’ this specifies the change or changes made to the model when performing counterfactual analysis. *E.g.* counterfactual = [{start="2020-4-01", type="trans_rate_fac", trans="S->E", factor="0.5"}] *start* – Gives the start time for the change (optional, set to ‘start’ by default). *end* – Gives the start time for the change (optional, set to ‘end’ by default). *type* – Denotes the type of change. This can either be “trans_rate_fac” to change the rate of a transition, or “efoi_fac” to change the external force of infection. *trans* – Specifies the transition on which the change applies. *geo* – Only apply to a particular geography (optional). *democats* – Only apply to a particular set of demographic categories (optional).
cutoff	Specifies the cut-off in the error function (‘abc’ mode). *E.g.* ./beepmbp inputfile="file.toml" mode="abc" nsample=200 cutoff=1
cutoff_frac	Specifies the fraction of accepted samples in the ‘abc’ mode. *E.g.* ./beepmbp inputfile="file.toml" mode="abc" nsample=200 cutoff_frac=0.002
datadir	The directory where the data files are stored. *E.g.* datadir= "examples/Data_Scotland”
data_tables	This denotes the files that contain the actual epidemiological data *E.g.* data_tables= [{type="transition", observation="C->D", geo="all", democats = "age", file="death.txt", start="2020-01-04", shift="0", units="weeks", weight=1.0}] *type* – Indicates the type of data • transition – This gives time series information about the number of individuals moving between two specified compartments. • population – This gives time series population numbers within a specified compartment. • marginal - This gives the total number of transitions within a time region, typically stratified by a demographic classification.

	observation – Describes what is observed. When the type is 'transition' or 'marginal' this is given by the initial compartment followed by "->" and then the final compartment (additionally, multiple transitions can be specified, *e.g.* "I->D,I->R", and the sum of them is used). For 'population' this is simply the compartment under measurement. *geo* – 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* – Set if the data is stratified by a given demographic classification (optional). *file* – The name of the file. This must be either a tab-separated '.txt' text file or a '.csv' file. *start* – Gives the start date for the data. *shift* – Allows for the data to be shifted in time. units – Gives the units in which the data exists. This can be "days" or "weeks". *weight* – This can give preferential weight when fitting the data (optional, and by default 1). A higher value implies that this particular data will be fit more stringently than others.
democats	Gives one or more other demographic categories. *E.g.* democats = [[{name="Sex", value="Male", sus="sus_M", sus_value="0.7"}, {value="Female", sus="sus_F", sus_value="1.3"}]] *name* – Gives the name for the demographic category. *value* – Gives the value of the category. *sus* – Defines a parameter giving the relative susceptibility. *sus_value* – The value for that parameter (under simulation).
efoi_factor	This gives the denominator when the external force of infection is specified (optional, by default set to infections per unit time per 100,000 individuals). *E.g.* efoi_factor = "100000"
efoi_spline	Defines the linear spline used to represent external force of infection. *E.g.* efoi_spline= [{ param="phi", value="908", bp="bp|phi_spline.txt", factor="factor|phi_spline.txt"}] See R_spline for property definitions. Also: *age_dist* - The fraction of individuals in different demographic 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).
end	The ending times for the simulation or inference (in days or as a date). *E.g.* end = "2020-12-31"
geo_mixing_ matrix	This gives a files used to define the geographic mixing between different regions. *E.g.* geo_mixing_matrix = "census_flow_data.txt"

geo_mixing_ modify	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 = [{ param="d_.", bp="bp|gmm.txt", value="value|gmm.txt", prior="Uniform(0.0,1)}]
geography	This sets the geography over which analysis is performed (optional). By default each of the lines in the 'areas' file represents a different area in the analysis. Not yet implemented…
infected_max	Sets the maximum number of infected individuals in the system (this can be used as a prior). *E.g.* infected_max = 100000
invT_final	Sets the inverse temperature for the posterior (used in 'pais' and 'mc3' modes). *E.g.* invT_final = 1.0
inputfile	Sets the input TOML file. *E.g.* inputfile="examples/SEIR.toml"
mode	This selects the mode of operation. *E.g.* mode = "sim" The following possibilities exist: *sim* – Simulates from the model. *multisim* – The model is simulated multiple times and ensemble generates outputs. *ppc* – Performs a posterior predictive check. *counter* – Performs counterfactual analysis. *abcmbp* – Performs inference using the ABC-MBP algorithm. *pais* – Performs inference using the PAIS-MBP algorithm. *abc* – Performs inference using the ABC algorithm. *abcsmc* – Performs inference using the ABC-SMC algorithm. *pmcmc* – Performs inference using the PMCMC algorithm. *mc3* – Performs inference using the MC^3 algorithm.
nburnin	Sets the number of 'burn-in' steps when performing MCMC (used in 'pmcmc' and 'mc3' modes). *E.g.* nburnin = 100 By default burn-in is set to a quarter of the number of samples 'nsample'
nchain	Sets the number of chains used when performing MC^3 in mode 'mc3' *E.g.* nchain = 20
ngeneration	Sets the number of generations (require in 'abcmbp', 'abcsmc' and 'pais' modes). *E.g.* ngeneration = 40
nparticle	The number of particles (required in 'abcmbp', 'pais' and 'pmcmc' modes)

	E.g. nparticle = 100
nquench	The number of time-step used in quenching the system when using the 'mc3' mode (optional, set to a half of nburnin by default). *E.g.* nquench = 100
nsample	Sets the number of samples to be generated (required in the 'abc', 'abcsmc', 'pmcmc' and 'mc3' modes). *E.g.* nsample = 200
nthin	Sets the thinning of samples (in 'pmcmc' 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). *E.g.* nthin = 10
outputdir	Gives the name of the output directory (optional, set to "Output" by default). *E.g.* outputdir = "Output_SEIR"
output_prop	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, 200 by default). *h* – A smoothness parameter when using kernel density estimation (optional, 10 by default).
quench_factor	Sets the rate of quenching in the 'pais' mode (optional, set to 0.5 by default such that on average half the particle each generation are discarded). *E.g.* quench_factor = "0.1"
propsize	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 kernel. *E.g.* propsize = "1.5"
pcc_start	The start time when performing a posterior predictive check (optional, set to 'start' by default'). *E.g.* pcc_start = "2020-03-01"
prob_reach	The probability of reaching 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 which is reached. *inf_trans* – If there are more than one infection transitions, this gives the transition on which the infectivity acts (optional).

region_effect	This is set if there is a regional effect incorporated at a particular geographical scale. *E.g.* region_effect = {geography="area", sigma="sigma", sigma_value="0.3"} *geography* – The geography over which the regional effect acts (optional, set to all areas by default). *sigma, sigma_value, sigma_prior* – The standard deviation in the regional effect.
R_spline	Defines the linear spline used to represent the reproduction number R. *E.g.* R_spline = [{ param="R|R_spline.txt", value="value|R_spline.txt", prior="Uniform(0.4,3)", bp="bp|R_spline.txt", smooth="0.4"}] *param* – The parameter names used to specify each spline point. *value* – The values for the parameters (optional). *prior* – The prior for the parameters (optional). *bp* – The times at which the break points occur. *smooth* – The level of smoothing used on the spline. *factor* – A multiplicative factor which multiplies the value of the parameter on each spline point (optional). *inf_trans* – If more than one infection transition exists in the model this specifies which one (optional, set to all transitions by default). *area* – Specifies the area on which the spline acts (optional, set to all areas by default). *factor_param* – The name of parameter which multiplies the whole spline by a factor (optional). *factor_value* –The value of that factor (optional). *factor_prior* – The prior on that factor (optional).
seed	This is a positive integer used to set the random seed. Note because the model is stochastic in nature then changing the seed will lead to a different simulated set of results. *E.g.* seed = 10
start	Sets the start time for the simulation or inference (in days or a date). *E.g.* start = "2020-01-01"
state_outputs	This allows for posterior graphs of the state to be plotted. *E.g.* state_outputs= [{type="transition", observation="S->E"}] *type* – Indicates the type of data • transition – This gives time series information about the number of individuals moving between two specified compartments. • population – This gives time series population numbers within a specified compartment. • marginal - This gives the total number of transitions within a time region, typically stratified by a demographic classification. *observation* – Describes what is observed. When the type is 'transition' or 'marginal' this is given by the initial compartment followed by "->" and then the final compartment (additionally, multiple transitions can be specified, *e.g.*

	"I->D,I->R", and the sum of them is used). For 'population' this is simply the compartment under measurement. *geo* – 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* – Set if the data is stratified by a given demographic classification (optional).
threshold	When data contains values that are below a threshold (*e.g.* a '*' represents 5 or less) this command specifies that threshold. *E.g.* threshold = 5
timeformat	This determines how times are represented in the initialisation and data files. *E.g.* timeformat = "year-month-day" The following possibilities exist: year-month-day – Dates use the format "2020-03-01" to represent 1st March. *number* – An integer is used to represent the time (*e.g.* the day number).
times	This is used to represent specific times of interest in the analysis. These can not only be used subsequent instead of dates (*e.g.* when specifying the break points in a spline), but are also places on the final time-based output graphs. *E.g.* times = [{name="Lockdown", date = "2020-03-23"}]
trans	Defines transitions between compartments. *E.g.* trans =[{from="S", to="E", dist="Infection"}, {from="E", to="I", dist="Erlang", k=2, mean="tE", mean_value="4"}, {from="I", to="R", dist="Exp", mean="tI", mean_value="4", prob="IR", prob_value="0.8"}, {from="I", to="D", dist="Exp", mean="tI", mean_value="4", prob="ID", prob_value="0.2"}] *from* – Gives the initial compartment. *to* – Gives the final compartment. *dist* – Gives the distribution in waiting time when going down the transition. This can either be "Infection" corresponding to when individuals become infected, "Exp" for exponential or "Erlang" for the Erlang distribution. *k* – For the Erlang distribution this sets the shape parameter (it must be a positive integer). *mean, mean_value, mean_prior* – Define the model parameter(s) used for the mean waiting time. *prob, prob_value, prob_prior* – Define the model parameter(s) used for the probability of branching down this transition (this must be set if more than one transitions leave a compartment). Note, if parameters are different for different demographic groups (*e.g.* ages), they can be set separately (see section).