Guide to using the code for the iPAR modelling framework

Stephen Catterall

28 March 2025

The transmission model

The code in this repository allows the user to implement the analyses described in the preprint https://www.biorxiv.org/content/10.1101/2024.11.08.622597v1. The code is split into two parts: simulation and inference. The simulation component generates stochastic realisations of the model, while the inference component performs Bayesian MCMC and outputs samples from the posterior distribution, given the provided disease outbreak data.The model assumed by the code is a slight generalisation of the model described in the preprint. The main difference is in the external transmission process. In the preprint, the background force of infection on a patch i is taken to be

$$\varepsilon s_i$$

where s_i is the susceptibility of patch i. In the code, the background force of infection on patch i is taken to be

$$\left(\varepsilon_1 e_{i,1} + \varepsilon_2 e_{i,2} + \varepsilon_3 e_{i,3} + \varepsilon_4 e_{i,4}^{-2\lambda} + \varepsilon_5 e_{i,5}^{-2\lambda} + \varepsilon_6 e_{i,6}^{-2\lambda}\right) s_i$$

where the Greek letters are parameters to be fitted to the data and $e_{i,k}$ are covariates associated with each patch i. Note that λ is the kernel parameter, the idea being that, for example, $e_{i,4}$ represents the distance from patch i to some external source of infection. The model described in the preprint is obtained as a nested submodel by setting $\varepsilon_k = 0$ for $k > 1$ and $e_{i,1} = 1$. The following sections describe how to implement this submodel.

Compiling the code

The code is written in the C programming language, so it needs to be compiled before use. Pre-compiled executables are available in the github repository for Windows 10 and Debian GNU/Linux 12. These executable can be used to reproduce the analyses decribed in the manuscript. However, if you wish to modify the code for other applications then it will be necessary to recompile it before use. Modification might be needed if you want to increase the number of covariates beyond the limit specified in the code. It would also be possible to change the kernel specification.

To compile the code on Windows 10, you will need to install a C compiler and have the GNU Scientific Library available. There are multiple ways of achieving this, but one option is to install the MSYS2 system https://www.msys2.org. This system will also be installed as part of the Rtools software which may already be installed for the purposes of developing R packages. If MSYS2 is installed in this way then the C compiler gcc should now be available for use. Open a MSYS2 terminal from the Windows start menu and install the GSL package using

```
pacman -S mingw-w64-x86_64-gsl
```

Now it should be possible to compile the code from the Windows command prompt using

```
gcc simulation_vi.c -lgsl -lgslcblas -o simulation_vi.exe
```

for the simulation code and

```
gcc inference_vi.c -lgsl -lgslcblas -o inference_vi.exe
```

for the inference code.

To compile on a Linux system, ensure that the C compiler gcc and the GNU Scientific Library are available, for details on the latter see https://www.gnu.org/software/gsl. It should now be possible to compile using commands such as

```
gcc simulation_vi.c -o simulation_vi -lm -lgsl -lgslcblas
```

and similarly for the inference code.

Inputs for the simulation code

The simulation code, in the folder `iPAR/simulation`, generates stochastic simulations from the iPAR model. The required inputs are as follows. **First**, there is a text file `inputs.txt` which contains several settings used by the simulation code. The file is formatted as follows. For each setting, there is a line in the file with the name of the setting. The setting itself is on the line following the name, and is either a number or a vector of numbers. If the file is not formatted in this way then an error will occur when the file is processed by the software. The settings are as follows.

- `Number_Simp` is the number of compositional covariates (land use categories), denoted L in the manuscript
- `Number_Eps` is the maximum permissible number of covariates used in the backgound transmission process (set to 6)
- `Number_Exp` is the number of non-compositional covariates for each patch, denoted K in the manuscript
- `nunits` is the number of patches in the modelled region
- `nsi` is the number of nonzero elements in the `subinttimes` vector (see below)
- `subinttimes` is the sequence of reporting times $t_0 = 0, t_1, \cdots, t_N$ in the manuscript
- `model` can be ignored and is always fixed at 1
- `numtk` is the number of changepoints in the transmsission rate (0 for the constant-in-time model)
- `tk` is the vector of changepoints, with the end time (`max(subinttimes)`) appended
- `maxdisp` is the maximum transmission distance in patch units, denoted d_{max} in the manuscript

- `numregions` can be ignored
- `regions` can be ignored
- `numfiles` is the number of input parameter files (see below)
- `numsamplesperfile` is the number of parameter sets per parameter file (see below)
- `burnin` is the number of burnin samples in each parameter file if using MCMC output
- `numsims` is the required number of simulations

Next, there are parameter files containing values of the model parameters to be used as inputs for the simulations. It is anticipated that the simulation code will be run in one of two ways. First, as a means of generating simulated outbreak data for a specific set of parameters. In this case, there is just a single parameter file `par_0.txt` (so `numfiles` is set to 1) containing a single parameter combination as a vector of numbers separated by spaces, all on a single line. The vector is ordered as follows.

- λ
- ρ
- ε (length `Number_Eps`)
- σ (length `Number_Simp`)
- γ (length `Number_Simp`)
- h (length `numtk`, parameters encoding the time-varying transmission function)
- σ' (length `Number_Exp`)
- γ' (length `Number_Exp`)
- some dummy numbers used for technical reasons (length `2*Number_Simp-1`)

The second use for the simulation code is to predict future spread via computation of posterior predictive distributions. In this case, we use a large number of samples from the posterior distribution as an input to the model, with one file per MCMC chain. There are usually five chains (so `numfiles` is set to 5), with the parameter files named `par_1.txt`, `par_2.txt`, `par_3.txt`, `par_4.txt` and `par_5.txt`. Each files contains `numsamplesperfile` lines, and each line contains a parameter vector with the same ordering as specified above. The first `burnin` lines of each file will be read in but will not be selected by the simulation code as samples from the posterior distribution.

Next, there is the covariates file `covariates.dat`. This is another text file. There are `nunits` lines and each line contains the covariates for a single patch. Usually some pre-processing is required in order to ensure that covariates obtained from spatial data are in the correct format. The covariate are given in the following order, separated by spaces.

- x coordinate of the patch
- y coordinate of the patch
- the `Number_Simp` compositional covariates
- the `Number_Eps` covariates used in the background transmission process (ignore these for the analyses in the manuscript)
- the `Number_Exp` non-compositional covariates

The code assumes that the coordinates of (the centroid, or some other reference point) of each patch are integers, more precisely they are integer multiples of a basic distance unit. If all the patches are square then this basic unit is the length of the side of one of these squares. This assumption about the coordinates is satisfied in the applications presented in the manuscript, and helps to speed up the kernel computations.

Finally, there is the initial conditions file `init.dat`. The first line of this text file contains the number `ninit`of initially infected patches. There are then a further `ninit` lines: each one contains the number, from `1` up to `ninit`, of an infected patch, with the ordering of the patches as specified in the covariates file.

NB On Linux there may be a problem with the software not processing the input files correctly because of differences in how Windows and Linux handle new lines in text files. If a text file created in Windows is not being processed correctly in Linux then it would be worth trying a command such as the following.

```
tr -d '\r' <oldinputfile.txt> newinputfile.txt
```

Running the simulation code

In the `simulation` folder in the repository there are some example input files for the simulation code. You can place the simulation executable in the same folder (either the pre-compiled file available in the repository or your own compiled version). Then just type `simulation_vi.exe` or `./simulation_vi` on Linux, to run the simulation code.

Outputs of the simulation code

When you run the simulation code, several output files are produced.

If the number of input parameter vectors was equal to one (the first mode of operation described above) then output files of the form `obsg_n.dat` are produced, where `n` denotes the number of the simulation starting from `0`. These files contains detailed information about each simulated outbreak. For the example input files provided, the number of simulations `numsims` has been set to `1` so there a single file `obsg_0.dat` corresponding to a single simulated outbreak. The file is a text file containing one line per patch, with the line ordering being the same as for the provided covariates file. Each line contains the following information.

- the x coordinate of the patch
- the y coordinate of the patch
- a dummy number
- the time category of the patch's infection time, which is denoted T_i in the manuscript (but note that this is set to `1000` if the patch remains uninfected at the end of the simulation)
- the actual time of the patch's infection, or `max(subinttimes)+1` if the patch remains uninfected at the end of the simulation

Other output files will be produced regardless of the number of input parameter vectors. These files provide summaries of the simulations in different ways. The file `riskspatial.txt` is used to create risk maps for future spread of infection. Each line in the file corresponds to a patch (ordered as in the covariates file). On each line is a vector of numbers that is the same length as the `subinttimes` vector. The n'th number is the proportion of simulations in which the patch was infected by the n'th time in the `subinttimes` vector. So, for example, the first number in a line is always `0` or `1` depending on whether the patch was initially infected or not.

The file `risktemporal.txt`summarises the number of infected patches at the times defined by the `subinttimes` vector. The file contains one line for each possible number of infected patches, starting at `0` and ending at `nunits`. On each line is a vector of numbers that is the same length as the `subinttimes` vector. The n'th number in the i'th line is the proportion of simulations in which `i-1` patches were infected at time `subinttimes[n]`. There is also a file `incidence.txt` which is similar in format but only counts new infections during each time interval defined by the `subinttimes` vector, rather than the cumulative total.

Having run the simulation code with the provided settings, you should now have some simulated outbreak data `obsg_0.dat` which can be used as an input for the inference algorithm. This is the component of the software that we will cover next.

Inputs for the inference code

The inference code, in the folder `iPAR/inference`, fits the iPAR model to outbreak data. The required inputs are as follows. **First**, there is a text file `parameters.txt` which contains several settings used by the inference code, in particular settings for the prior and proposal distributions for model parameters. The file is formatted as follows. For each setting, there is a line in the file with the name of the setting. The setting itself is on the line following the name, and is either a number or a vector of numbers. If the file is not formatted in this way then an error will occur when the file is processed by the software. The settings are as follows.

- `Number_Simp` is the number of compositional covariates (land use categories), denoted L in the manuscript
- `Number_Eps` is the maximum permissible number of covariates used in the backgound transmission process (set to 6)
- `Number_Exp` is the number of non-compositional covariates for each patch, denoted K in the manuscript
- `nunits` is the number of patches in the modelled region
- `nsi` is the number of elements in the `subinttimes` vector (see below)
- `subinttimes` is the sequence of reporting times $t_0 = 0, t_1, \cdots, t_N$ in the manuscript
- `model` can be ignored and is always fixed at 1
- `numtk` is the number of changepoints in the transmsission rate (0 for the constant-in-time model)
- `tk` is the vector of changepoints, with the end time (`max(subinttimes)`) appended

- `maxdisp` is the maximum transmission distance in patch units, denoted d_{max} in the manuscript
- `burnin` is the number of MCMC burnin iterations (we suggest 10000, but this is set to 100 in the example file provided for testing purposes)
- `mcmc` is the number of MCMC iterations following burnin (we suggest 90000, but this is set to 900 in the example file provided for testing purposes)

As can be seen in the file provided, there follows a list of settings for each parameter (and each component for vector parameters), starting with `lambda`. The settings for each parameter are as follows.

- `initial` is a vector of five initial values for the parameter, one for each of five MCMC chains
- `propsd` is a vector of standard deviations for the Gaussian proposal distribution for the parameter, these are initial values that may change during the burnin phase
- `prior_e` is `1` if an exponential prior is to be used, `0` for a uniform prior
- `prior_r` is the rate for an exponential prior
- `prior_l` is the lower limit for a uniform prior
- `prior_u` is the upper limit for a uniform prior

Note that, for components of compositional parameters, some of these values are dummy numbers that are not actually used in the algorithm. This is because such parameters are always assumed to have a uniform prior distribution on the corresponding simplex.

Next, there is the covariates file `covariates.dat` which has exactly the same format as described above for the simulation code.

Finally, there is the outbreak data file `data.dat`. In this file each line contains two numbers. The first line is special and contains the number of observations (the number of patches recorded as infected) followed by a dummy number. There are then additional lines, one for each observation. The first number is the number of the patch observed as infected; numbers range from `0` to `nunits-1`, with patches ordered as in the covariates file. The second number is the time category of the patch's infection time, which is denoted T_i in the manuscript.

Running the inference code

In the `inference` folder in the repository there are some example input files for the inference code. You can place the inference executable in the same folder (either the pre-compiled file available in the repository or your own compiled version). Then just type `inference_vi.exe -r 1` or `./inference_vi -r 1` on Linux, to run the inference code with `r=1`, meaning that this runs inference for chain 1. To run inference for all five chains, you could use a schell scipt such as the following.

```
for i in $(seq 1 1 5)
do
./inference_vi.exe -r $i
done
```

You can use the simulated data generated by the simulation code, `obsg_0.dat` as an input for the inference code. Copy the file into the inference folder. Then apply the following R code.

```
covd<-matrix(scan("covariates.dat"),ncol=14,byrow=TRUE)
x<-as.integer(covd[,1])

obsfile=matrix(scan("obsg_0.dat"),ncol=5,byrow=TRUE)
obs=obsfile[,1:4]
obs<-obs[obs[,4]<1000,]
xobs<-obs[,1]
yobs<-obs[,2]
gobs<-obs[,4]
iobs<-rep(0,length(xobs))
for(i in 1:(length(iobs)))
{
  iobs[i]=(0:(length(x)-1))[x==xobs[i] & y==yobs[i]]
}
nobs=length(gobs)

iobs<-c(nobs,iobs)
gobs<-c(nobs,gobs)

data<-cbind(iobs,gobs)

write.table(data,row.names=F,col.names=F,file="data.dat")
```

Now it should be possible to run the inference code on the newly created `data.dat`.

Outputs of the inference code

The output of the inference algorithm consists of samples from the posterior distribution of the model parameters. The samples of the parameters from chain 1 are contained in file `par_1.txt`, while the samples of the infection times are contained in `parlatent_1.txt`. The outputs from the other chains are named similarly. In `par_1.txt` there is one line for each MCMC sample, including the burnin samples. Each line is a parameter vector ordered as follows.

- λ
- ρ
- ε (length `Number_Eps`)
- σ (length `Number_Simp`)
- γ (length `Number_Simp`)
- h (length `numtk`, parameters encoding the time-varying transmission function)
- σ' (length `Number_Exp`)
- γ' (length `Number_Exp`)
- the likelihood (not technically a parameter but useful to know)
- isometric logratio transform of σ (length `Number_Simp-1`)

- isometric logratio transform of γ (length `Number_Simp-1`)

To obtain parameter estimates it is recommended that the output from all chains is amalgamated, with burnin samples removed. If, instead, the samples are to be used as inputs for simulations of the posterior predictive distribution then the files `par_1.txt` etc. can be used directly as inputs for the simulation code.