The EpiFusion Analysis Framework for joint phylodynamic and epidemiological analysis of outbreak characteristics

## Abstract

The fields of epidemiology and phylodynamics share the ultimate goal of disease control, yet employ distinct approaches. Both use computation and mathematics to model disease spread, but concepts, methodologies and data employed by each differ in ways that confer complementary strengths and different areas of weakness. Thus, there is increasing interest in both fields in methods that cohesively combine principles and data from both fields into unified models. We recently introduced EpiFusion, a model for joint inference of outbreak characteristics using phylogenetic and case incidence data via particle filtering, and demonstrated its usage to infer the effective reproduction number of simulated and real outbreaks. Here we provide a series of vignettes demonstrating data analysis using the EpiFusion analysis framework, consisting of the R package EpiFusionUtilities and the java program in which the model is implemented – EpiFusion.

## Introduction

## Methods

### Operation

EpiFusion is implemented as a free open source Java software (version 8 or later) and can be used as a command line tool or from within the EpiFusionUtilities R package. The latest stable version of the program is available for download under Releases on the project Github repository. The source code for the latest development version is also available at this repository, to facilitate users who wish to clone the repository and compile the program from source. Consider carefully which directory you place the EpiFusion file in, as every time you wish to use it you’ll have to call it with its file path. You can also call EpiFusion from any working directory on your system by creating a symbolic link *(Appendix 1)*.

EpiFusionUtilities is implemented as an open source R (version 3.5.0 or later) package and is available to install from Github using devtools:

# install from Github  
devtools::install\_github("https://github.com/ciarajudge/EpiFusionUtilities")

The recommended EpiFusion workflow includes data processing and parameterisation in R using EpiFusionUtilities (this may also be done manually), followed by running from the EpiFusionUtilities function run\_epifusion() or by calling EpiFusion directly from the command line, and finally, parsing the results from output files into familiar R objects using the EpiFusionUtilities functions load\_raw\_epifusion() and extract\_posterior\_epifusion(). The key steps of the EpiFusion Framework workflow are outlined below in brief in the ‘Implementation’ section.

### Implementation

#### Parameterisation / Data Processing

##### EpiFusion XML

EpiFusion uses input files written in eXtensible Markup Language (XML) to provide all data and parameterisation to the program. These files contain Loggers, Data, Analysis, Model, Parameters and Priors sections where various aspects of the model and analysis may be specified *(Table 1)*. A full breakdown of the options available within each section is included in the Supplementary Information *(Appendix 2)*.

Table 1 Main sections of EpiFusion XML parameter file structure

| EpiFusion XML Section | Description |
| --- | --- |
| Loggers | Provides detail on the program output, including specifying the file path of the output folder that should be created, and the frequency at which the program logs the state of the MCMC to the output files (and prints to the console) |
| Data | Provides the case incidence data and/or a phylogenetic tree (either within the XML document or by providing file paths to files containing the data) |
| Analysis | Parameterises the method for fitting beta, the force of infection |
| Model | Allows further customisation of the EpiFusion model structure - currently only specification of the epidemiological observation model is available |
| Parameters | Contains many assorted parameters for the model, for example number of MCMC steps per chain, number of MCMC chains, number of particles in the particle filter etc. |
| Priors | Prior specification for parameters to be fit via particle MCMC |

##### Assembling parameter XML files

EpiFusion XML files may either be populated manually using the many templates available at the EpiFusion Github repository, or created using a number of useful functions in the EpiFusionUtilities package.

The first step of processing data for EpiFusion input is to select an ‘index date’. The index date is the earliest possible date of origin of the outbreak (i.e. day 0), and is provided to the processing functions to enable the case and incidence data to be rooted in numerical time units. All times in EpiFusion input and output will be in relation to this date and measured in days.

index\_date <- as.Date("2024-01-01")

If the date of outbreak origin is not being inferred (inferTimeOfIntroduction is false in the analysis block of the XML), all trajectory samples will assume that the outbreak originated with one infected individual on the index date. If the date of outbreak origin is being inferred, the model will fit the date at which the first individual is infected, thus trajectory samples may begin with 0 infected until the date of origin which the MCMC process has sampled.

To prepare a tree or tree posterior for EpiFusion, pass an S3 phylo or multiPhylo object in R to the prepare\_epifusion\_tree() function. This function processes a phylogenetic tree (or trees) and writes to a file, which you can specify in the arguments of the function (the default is ‘./processedtree.tree’). It is also necessary to pass the date of the last sample in the tree(s).

prepare\_epifusion\_tree(tree,  
 index\_date,  
 last\_sequence,  
 "Data/Processed/processed\_fixed\_tree.tree")

To generate an EpiFusion XML file from within an R session, the generate\_epifusion\_XML() function may be used. This function populates a template xml file with the provided data, and has default settings for most parameters and priors which can also be changed by providing new values in the arguments of the function. For example below, we pass a case incidence data frame and tree to the function, specify that we will sample from the MCMC chain every 100 steps, set our output folder path to ‘output\_files’, and adjust the number of particles in the particle filter to 300. This creates a file in our working directory ‘epifusion\_input.xml’ which should be ready to pass to EpiFusion.

logger\_information <- list(fileBase = "output\_files", logEvery = 100)  
parameters\_to\_adjust <- list(numParticles = 300)  
  
generate\_epifusion\_XML(tree = "Data/Processed/processed\_fixed\_tree.tree",  
 case\_incidence = case\_incidence,  
 index\_date = index\_date,  
 loggers = loggers,  
 parameters = parameters,  
 xml\_filepath = "epifusion\_input.xml")

#### Running EpiFusion

EpiFusion can be run directly from the command line by calling an executable Java Archive (JAR) file using the following syntax. Here EpiFusion.jar is the file path to the executable file (in this example, the file is present in the working directory) and epifusion\_input.xml is the file path to the parameter XML file (also present in the working directory for this example):

java -jar EpiFusion.jar epifusion\_input.xml

Alternatively, it is possible to run EpiFusion from inside an R session with the EpiFusionUtilities function [run\_epifusion()](https://github.com/ciarajudge/EpiFusionUtilities/wiki/run_epifusion()). (an installation of Java is still required, but this option may be slightly more intuitive for some users).

run\_epifusion("epifusion\_input.xml")

#### Interpreting Output

EpiFusion creates a directory within the working directory that corresponds to the file path of the ‘fileBase’ parameter in your EpiFusion xml file. For each MCMC chain, EpiFusion will create the following output files:

* betas: csv file where each row is a trajectory of rate beta sampled from the MCMC
* trajectories: csv file where each row is a daily infection trajectory sampled from the MCMC
* params: txt file where each column is an MCMC parameter, and each row is an MCMC sample
* likelihoods: txt file of the posterior likelihoods from each MCMC step
* acceptance: txt file where each line logs the acceptance rate of steps between MCMC samples
* completed: txt file where each line logs if the particle filter step was completed or quit due to particle depletion
* cuminfections: txt file where each row is a trajectory of cumulative infections per day sampled from the MCMC
* positivetests: (only for combined or epi-only analyses): csv file where each row is simulated case incidence by the model which was compared to the observed case incidence

EpiFusion will also save a copy of the parameter file used to the output folder, so you can remember exactly what parameters were used, and a file called ‘timings.txt’ with the runtime in nanoseconds.

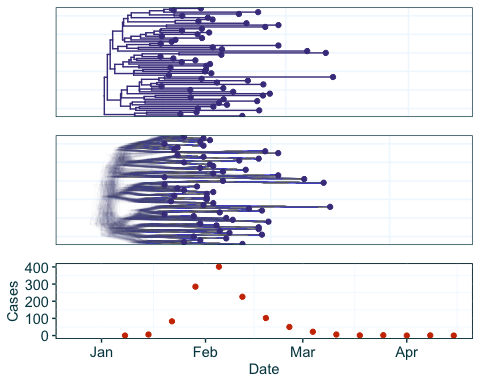
It is possible to process this raw output manually, but EpiFusionUtilities provides a number of helpful functions to do this neatly from within R. The following functions load the raw output into an R object, plot the likelihood trace for each MCMC chain to enable inspection to decide what proportion of samples from each chain to discard as burn-in, and finally extract the posterior samples from each chain and combine them into a single posterior while assessing convergence.

raw\_output <- load\_raw\_epifusion("output\_files/")  
plot\_likelihood\_trace(raw\_output)  
full\_posterior <- extract\_posterior\_epifusion(raw\_output, 0.1)

## Use Case - Full EpiFusion Framework Workflow

### Description of the simulated dataset

Below we demonstrate the implementation of EpiFusion and EpiFusionUtilities to analyse data from a small simulated outbreak. An outbreak trajectory, and resulting weekly case incidence and a phylogenetic tree of simulated sequences were generated using ReMaster, and genomic sequences were simulated in R from the phylogenetic tree using the function simSeq() from the R package phangorn. These sequences were used to generate a tree posterior using BEAST 2.7.3 with a Birth Death Skyline model, under a strict clock and JC69 substitution model. Further information on the BEAST specification is included in the Supplementary Information *(Appendix 4)*. The date of origin of the outbreak was arbitrarily chosen as January 1st 2024. The final resulting data inputs for analysis in EpiFusion consisted of a file with a fixed time-scaled phylogenetic tree, a tree posterior file generated from sequences simulated from the outbreak, and a csv file with dated counts of weekly incidence. These raw data files are provided in the article repository ‘xxx (need to upload to github)’, where the code below is provided to fully replicate this use case example.



Data from a simulated outbreak using ReMaster and BEAST 2.7.3

### Data Preparation

First we load and inspect the data. We will run two EpiFusion analyses in this example - one with the fixed phylogenetic tree, and one where we account for phylogenetic uncertainty by using the tree posterior as the data for EpiFusion. The case incidence can be read into the program as a csv file from Data/Raw/weekly\_incidence.csv, and the Date column coerced to Date format using the mutate and as.Date functions. The fixed tree is in the Data/Raw/fixed\_tree.tree file in newick string format, so we read it using the read.tree function of the ape package. The tree posterior generated by BEAST is in the Data/Raw/tree\_posterior.trees and is in nexus format, so can be loaded using the read.nexus function of the ape package.

case\_incidence <- read.csv("Data/Raw/weekly\_incidence.csv") %>%  
 mutate(Date = as.Date(Date))  
fixed\_tree <- read.tree("Data/Raw/fixed\_tree.tree")  
tree\_posterior <- sample(read.nexus("Data/Raw/tree\_posterior.trees")[2000:10000], 200) # Discard burn in, randomly sample 200  
  
print(case\_incidence[1:5,])

## Cases Date  
## 1 0 2024-01-08  
## 2 6 2024-01-15  
## 3 83 2024-01-22  
## 4 285 2024-01-29  
## 5 401 2024-02-05

print(fixed\_tree)

##   
## Phylogenetic tree with 59 tips and 58 internal nodes.  
##   
## Tip labels:  
## sequence1|2024-01-25, sequence2|2024-01-30, sequence3|2024-02-12, sequence4|2024-01-30, sequence5|2024-02-04, sequence6|2024-02-17, ...  
## Node labels:  
## node\_1, node\_4, node\_14, node\_97, node\_106, node\_158, ...  
##   
## Rooted; includes branch lengths.

print(tree\_posterior)

## 200 phylogenetic trees

Next we set two date objects: the ‘index date’, or the earliest date from which we will model the outbreak origin (this is most relevant for the second of the two analyses we run here, where we also infer the time of introduction or outbreak origin), and the date sampling of the last observed sequence from the dataset. While we know through the simulation process that the outbreak origin was the 1st of January 2024, it is good practice to set the index date to some time before when we suspect the outbreak originated, to allow the program freedom to infer the outbreak origin through a range of possible values.

index\_date <- as.Date("2023-12-15")  
last\_sequence <- as.Date("2024-03-10")

To prepare the tree objects for EpiFusion we can use the prepare\_epifusion\_tree function from EpiFusionUtilities. This function processes the tree(s) for input to EpiFusion and writes them to the provided file path. In the case where a single fixed tree (our first example) is provided to this function it also returns the processed tree as an R phylo object, which here we reassign to the variable fixed\_tree.

fixed\_tree <- prepare\_epifusion\_tree(fixed\_tree, index\_date, last\_sequence, "Data/Processed/processed\_fixed\_tree.tree")  
prepare\_epifusion\_tree(tree\_posterior, index\_date, last\_sequence, "Data/Processed/processed\_tree\_posterior.tree")

### Definition of parameters

We will create an EpiFusion XML file using the generate\_epifusion\_xml function from EpiFusionUtilities. This function populates the below XML template with our data and creates a new file, and other arguments to this function can adjust parameters from their default value.

<?xml version="1.0" encoding="UTF-8"?>  
<EpiFusionInputs>  
 <loggers>  
 <fileBase>FILESTEM</fileBase>  
 <logEvery>10</logEvery>  
 </loggers>  
 <data>  
 <incidence>  
 <incidenceVals>INCIDENCE</incidenceVals>  
 <incidenceTimes type="exact">INCIDENCETIMES</incidenceTimes>  
 </incidence>  
 <tree>  
 <treePosterior></treePosterior>  
 </tree>  
 <epicontrib>0.5</epicontrib>  
 <changetimes>0</changetimes>  
 </data>  
 <analysis>  
 <type>looseformbeta</type>  
 <startTime>null</startTime>  
 <endTime>null</endTime>  
 <inferTimeOfIntroduction>false</inferTimeOfIntroduction>  
 </analysis>  
 <model>  
 <epiObservationModel>poisson</epiObservationModel>  
 </model>  
 <parameters>  
 <epiOnly>false</epiOnly>  
 <phyloOnly>false</phyloOnly>  
 <numParticles>200</numParticles>  
 <numSteps>2000</numSteps>  
 <numThreads>8</numThreads>  
 <numChains>4</numChains>  
 <stepCoefficient>0.05</stepCoefficient>  
 <resampleEvery>7</resampleEvery>  
 <segmentedDays>true</segmentedDays>  
 <samplingsAsRemovals>1</samplingsAsRemovals>  
 <pairedPsi>false</pairedPsi>  
 </parameters>  
 <priors>  
 <gamma>  
 <stepchange>false</stepchange>  
 <disttype>TruncatedNormal</disttype>  
 <mean>0.143</mean>  
 <standarddev>0.05</standarddev>  
 <lowerbound>0.0</lowerbound>  
 </gamma>  
 <psi>  
 <stepchange>false</stepchange>  
 <disttype>TruncatedNormal</disttype>  
 <mean>0.001</mean>  
 <standarddev>0.0005</standarddev>  
 <lowerbound>0.0</lowerbound>  
 </psi>  
 <phi>  
 <stepchange>false</stepchange>  
 <disttype>TruncatedNormal</disttype>  
 <mean>0.02</mean>  
 <standarddev>0.01</standarddev>  
 <lowerbound>0.0</lowerbound>  
 </phi>  
 <initialBeta>  
 <stepchange>false</stepchange>  
 <disttype>Uniform</disttype>  
 <min>0.3</min>  
 <max>0.8</max>  
 </initialBeta>  
 <betaJitter>  
 <stepchange>false</stepchange>  
 <disttype>Uniform</disttype>  
 <min>0.001</min>  
 <max>0.05</max>  
 </betaJitter>  
 </priors>  
</EpiFusionInputs>

First we will generate an EpiFusion XML using the fixed tree we prepared with the prepare\_epifusion\_tree function. First we will make lists of the various parts of the XML file we wish to override from the default. For example, the below code represents the loggers chunk in the default XML:

<loggers>  
 <fileBase>FILESTEM</fileBase>  
 <logEvery>10</logEvery>  
 </loggers>

We will make a list in R that we will later pass to the loggers argument of the generate\_epifusion\_xml function to specify our output folder filepath as Results/fixed\_tree and sample from the MCMC chain every 5 steps:

loggers <- list(fileBase = "Results/fixed\_tree", logEvery = 5)

In this example we are happy with the other parameters and priors in the default XML, so we can generate the XML file Data/EpiFusion\_XMLs/fixed\_tree\_inputfile.xml with the following code:

generate\_epifusion\_XML(tree = "Data/Processed/processed\_fixed\_tree.tree",  
 case\_incidence = case\_incidence,  
 index\_date = index\_date,  
 loggers = loggers,  
 xml\_filepath = "Data/EpiFusion\_XMLs/fixed\_tree\_inputfile.xml")

Next we will generate the XML file for the analysis using the tree posterior. We again specify adjustments to the loggers chunk, and make some other changes. We adjust the method for fitting beta over time from a random walk to linear splines between resampling points, and switch on the inference of ‘Time of Introduction’ or ‘Outbreak Origin’ in the analysis block.

loggers <- list(fileBase = "Results/tree\_posterior\_and\_TOI", logEvery = 5)  
analysis <- list(type = "linearsplinebeta", inferTimeOfIntroduction = "true")

Because we are introducing inference of the outbreak origin, it is necessary to set a prior for the value in terms of days from the index\_date. The prior should take advantage of one of the discrete distributions available for prior parameterisation in EpiFusion. For the prior block only, it is not necessary to make targeted adjustments; i.e. any adjustments to the prior block result in it becoming necessary to respecify all priors.

priors <- list(gamma = list(stepchange = "false",  
 disttype = "TruncatedNormal",  
 mean = 0.15,  
 standarddev = 0.03,   
 lowerbound = 0.0),  
 psi = list(stepchange = "false",  
 disttype = "TruncatedNormal",  
 mean = 0.001,  
 standarddev = 0.0004,   
 lowerbound = 0.0),  
 phi = list(stepchange = "false",  
 disttype = "TruncatedNormal",  
 mean = 0.02,  
 standarddev = 0.01,   
 lowerbound = 0.0),  
 initialBeta = list(stepchange = "false",  
 disttype = "Uniform",  
 min = 0.3,  
 max = 0.8),  
 betaJitter = list(stepchange = "false",  
 disttype = "Uniform",  
 min = 0.001,  
 max = 0.1,   
 lowerbound = 0.0),  
 outbreakOrigin = list(stepchange = "false",  
 disttype = "UniformDiscrete",  
 min = 0,  
 max = 30,   
 lowerbound = 0.0))

Now we can generate an XML file for our tree posterior analysis by passing our data, parameter lists, and destination filepath to generate\_epifusion\_xml:

generate\_epifusion\_XML(tree = "Data/Processed/processed\_tree\_posterior.tree",  
 case\_incidence = case\_incidence,  
 index\_date = index\_date,  
 loggers = loggers,  
 analysis = analysis,  
 priors = priors,  
 xml\_filepath = "Data/EpiFusion\_XMLs/tree\_posterior\_inputfile.xml")

### Running EpiFusion

To run EpiFusion for the fixed tree example, we will use the run\_epifusion function from EpiFusionUtilities to run the program within our R session:

run\_epifusion("Data/EpiFusion\_XMLs/fixed\_tree\_inputfile.xml")

Alternatively, it is possible to run the program via the command line in a shell terminal with the following code, which we will demonstrate for the tree posterior example:

java -jar EpiFusion.jar "Data/EpiFusion\_XMLs/tree\_posterior\_inputfile.xml"

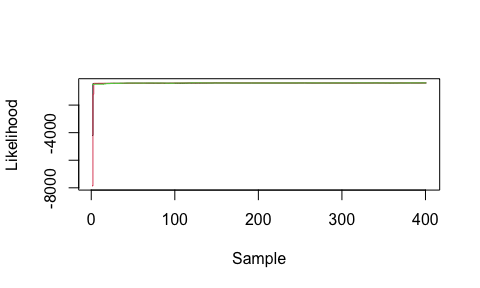
This code assumes there is an EpiFusion jar file in the working directory (which is provided in the Github repository with the rest of the raw data and code used in this Use Case).

### Parsing the output

We will primarily use the fixed tree example to demonstrate the process of parsing through EpiFusion output, however it the same commands can be used for parsing any EpiFusion output, including the tree posterior example. Later we will parse the tree posterior example to examine the inferred time of introduction and the effect of using the tree posterior on the results.

First we will use the load\_raw\_epifusion and plot\_likelihood\_trace functions to import the full raw results and examine the likelihood trace. This allows us to check for convergence and help to identify what proportion of each chain to discard as burn-in.

raw\_output\_fixed <- load\_raw\_epifusion("Results/fixed\_tree/")  
plot\_likelihood\_trace(raw\_output\_fixed)



Likelihood trace of the MCMC chains run in an EpiFusion model

Next we can discard the burn-in from each MCMC chain and combine all chains into a combined posterior using the extract\_posterior\_epifusion function which takes a raw EpiFusion object and the proportion of each chain to discard as burn-in as its arguments.

parsed\_output\_fixed <- extract\_posterior\_epifusion(raw\_output\_fixed, 0.1)  
str(parsed\_output\_fixed, max.level = 2)

## List of 5  
## $ infection\_trajectories:List of 3  
## ..$ mean\_infection\_trajectory : Named num [1:124] 0 1.3 1.68 1.94 2.21 ...  
## .. ..- attr(\*, "names")= chr [1:124] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ infection\_trajectory\_hpdintervals:List of 3  
## ..$ infection\_trajectory\_samples :'data.frame': 1086 obs. of 124 variables:  
## $ rt\_trajectories :List of 3  
## ..$ mean\_rt\_trajectory : Named num [1:124] 2.53 2.45 2.36 2.28 2.23 ...  
## .. ..- attr(\*, "names")= chr [1:124] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ rt\_trajectory\_hpdintervals:List of 3  
## ..$ rt\_trajectory\_samples :'data.frame': 1086 obs. of 124 variables:  
## $ parameters :List of 5  
## ..$ gamma :List of 3  
## ..$ psi :List of 3  
## ..$ phi :List of 3  
## ..$ initialBeta:List of 3  
## ..$ betaJitter :List of 3  
## $ fitted\_epi\_cases :List of 3  
## ..$ mean\_fitted\_epi\_cases : Named num [1:15] 0.236 12.858 84.748 297.844 414.495 ...  
## .. ..- attr(\*, "names")= chr [1:15] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ fitted\_epi\_cases\_hpdintervals:List of 3  
## ..$ fitted\_epi\_cases\_samples :'data.frame': 1086 obs. of 15 variables:  
## $ cumulative\_infections :List of 3  
## ..$ mean\_cuminfection\_trajectory : Named num [1:124] 0 0.305 0.681 1.156 1.662 ...  
## .. ..- attr(\*, "names")= chr [1:124] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ cuminfection\_trajectory\_hpdintervals:List of 3  
## ..$ cuminfection\_trajectory\_samples :'data.frame': 1086 obs. of 124 variables:

### Plotting and examining the output

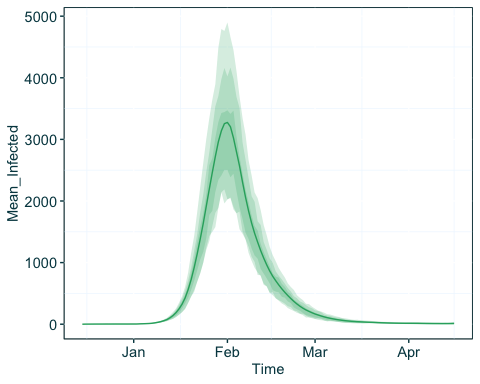
The extracted posterior object from the extract\_posterior\_epifusion function contains mean and HPD intervals of increasing width for infection, Rt, cumulative infection and fitted epidemiological case trajectories. The trajectory\_table function can parse these into a convenient table structured to be suitable for plotting with ggplot2.

traj\_table <- trajectory\_table(parsed\_output\_fixed, index\_date)  
head(traj\_table, n = 3)

## Time Mean\_Infected Lower95\_Infected Upper95\_Infected Lower88\_Infected  
## T\_0 2023-12-15 0.000000 0 0 0  
## T\_1 2023-12-16 1.304788 1 2 1  
## T\_2 2023-12-17 1.681400 1 3 1  
## Upper88\_Infected Lower66\_Infected Upper66\_Infected Mean\_Rt Lower95\_Rt  
## T\_0 0 0 0 2.526726 1.570177  
## T\_1 2 1 1 2.449210 1.306250  
## T\_2 3 1 2 2.362381 1.000285  
## Upper95\_Rt Lower88\_Rt Upper88\_Rt Lower66\_Rt Upper66\_Rt  
## T\_0 3.506708 1.652480 3.242646 1.958633 2.914854  
## T\_1 3.586019 1.537037 3.413762 1.943275 3.093608  
## T\_2 3.490892 1.384167 3.375482 2.075013 3.291919  
## Mean\_CumulativeInfections Lower95\_CumulativeInfections  
## T\_0 0.0000000 0  
## T\_1 0.3047882 0  
## T\_2 0.6813996 0  
## Upper95\_CumulativeInfections Lower88\_CumulativeInfections  
## T\_0 0 0  
## T\_1 1 0  
## T\_2 2 0  
## Upper88\_CumulativeInfections Lower66\_CumulativeInfections  
## T\_0 0 0  
## T\_1 1 0  
## T\_2 2 0  
## Upper66\_CumulativeInfections  
## T\_0 0  
## T\_1 0  
## T\_2 1

We will use this table with ggplot functions to plot and inspect the inferred infection trajectories:

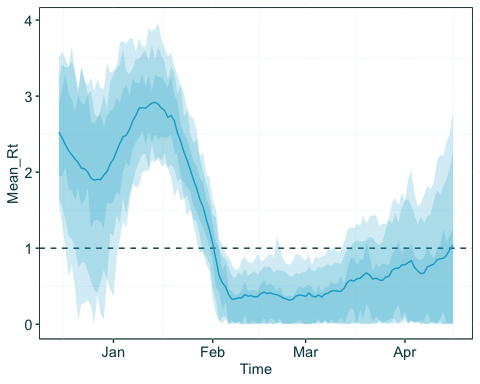
ggplot(traj\_table, aes(x = Time)) +  
 geom\_line(aes(y = Mean\_Infected), col = "#2aac6d") +  
 geom\_ribbon(aes(ymin = Lower95\_Infected, ymax = Upper95\_Infected), fill = "#2aac6d", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower88\_Infected, ymax = Upper88\_Infected), fill = "#2aac6d", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower66\_Infected, ymax = Upper66\_Infected), fill = "#2aac6d", alpha = 0.2) +  
 lshtm\_theme()



Inferred infection trajectories using a combined EpiFusion model and a fixed tree, plotted with ggplot2

We can use very similar code to plot the inferred R(t) trajectories:

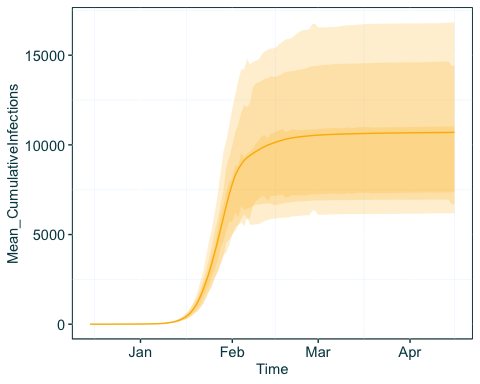
ggplot(traj\_table, aes(x = Time)) +  
 geom\_hline(yintercept = 1, linetype = 2, col = "#01454f") +  
 geom\_line(aes(y = Mean\_Rt), col = "#00abce") +  
 geom\_ribbon(aes(ymin = Lower95\_Rt, ymax = Upper95\_Rt), fill = "#00abce", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower88\_Rt, ymax = Upper88\_Rt), fill = "#00abce", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower66\_Rt, ymax = Upper66\_Rt), fill = "#00abce", alpha = 0.2) +  
 lshtm\_theme()



Inferred R(t) trajectories using a combined EpiFusion model and a fixed tree, plotted with ggplot2

Here we can see that the inferred R(t) from our index date of 15th December 2023 to the date of the true outbreak origin (January 1st 2024) is characterised by uncertainty in the R(t) estimates, which is to be expected. Finally we will plot the cumulative infections:

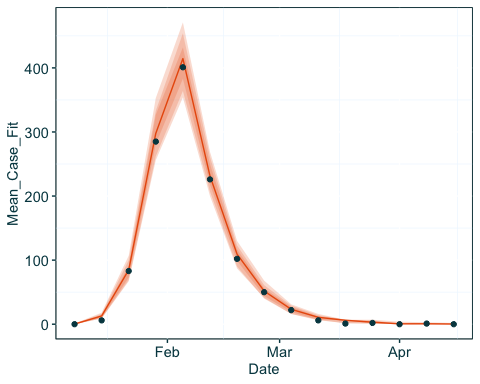
ggplot(traj\_table, aes(x = Time)) +  
 geom\_line(aes(y = Mean\_CumulativeInfections), col = "#fbb800") +  
 geom\_ribbon(aes(ymin = Lower95\_CumulativeInfections, ymax = Upper95\_CumulativeInfections), fill = "#fbb800", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower88\_CumulativeInfections, ymax = Upper88\_CumulativeInfections), fill = "#fbb800", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower66\_CumulativeInfections, ymax = Upper66\_CumulativeInfections), fill = "#fbb800", alpha = 0.2) +  
 lshtm\_theme()



Inferred cumulative infection trajectories using a combined EpiFusion model and a fixed tree, plotted with ggplot2

As this has been a combined analysis that has used case incidence data, it is possible to examine the fit of the case incidence simulated within the model to the provided data. We already have the case incidence data loaded from the data preparation stage, so we can add the mean and HPD intervals of the fit to the existing table.

epi\_data\_table <- case\_incidence %>%  
 mutate(Mean\_Case\_Fit = parsed\_output\_fixed$fitted\_epi\_cases$mean\_fitted\_epi\_cases,  
 Lower95\_Cases = parsed\_output\_fixed$fitted\_epi\_cases$fitted\_epi\_cases\_hpdintervals$HPD0.95$Lower,  
 Upper95\_Cases = parsed\_output\_fixed$fitted\_epi\_cases$fitted\_epi\_cases\_hpdintervals$HPD0.95$Upper,  
 Lower88\_Cases = parsed\_output\_fixed$fitted\_epi\_cases$fitted\_epi\_cases\_hpdintervals$HPD0.88$Lower,  
 Upper88\_Cases = parsed\_output\_fixed$fitted\_epi\_cases$fitted\_epi\_cases\_hpdintervals$HPD0.88$Upper,  
 Lower66\_Cases = parsed\_output\_fixed$fitted\_epi\_cases$fitted\_epi\_cases\_hpdintervals$HPD0.66$Lower,  
 Upper66\_Cases = parsed\_output\_fixed$fitted\_epi\_cases$fitted\_epi\_cases\_hpdintervals$HPD0.66$Upper)  
  
  
ggplot(epi\_data\_table, aes(x = Date)) +  
 geom\_line(aes(y = Mean\_Case\_Fit), col = "#e95b0d") +  
 geom\_ribbon(aes(ymin = Lower95\_Cases, ymax = Upper95\_Cases), fill = "#e95b0d", alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower88\_Cases, ymax = Upper88\_Cases), fill = "#e95b0d",alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower66\_Cases, ymax = Upper66\_Cases), fill = "#e95b0d", alpha = 0.2) +  
 geom\_point(aes(y = Cases), col = "#01454f") +  
 lshtm\_theme()



Fit of observed epidemiological cases to simulated cases by the EpiFusion model, plotted with ggplot2

Finally we can examine the posteriors of the MCMC parameters. The posterior extraction process uses the R package stable.GR to perform gelman-rubin convergence tests on each parameter, and estimate the effective sample sizes of each:

print(parsed\_output\_fixed$parameters$gamma$rhat)

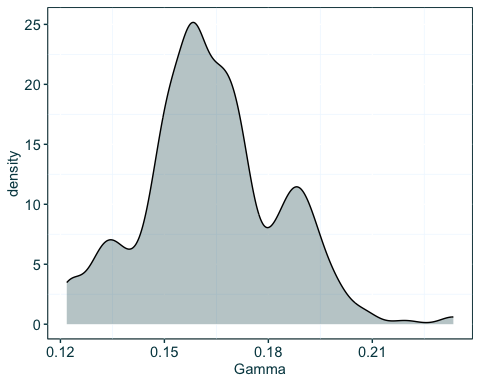
## [1] 1.017957

print(parsed\_output\_fixed$parameters$gamma$ess)

## [1] 77

We can also view the posterior density by plotting the samples of a parameter from the MCMC:

ggplot(data = data.frame(Gamma = parsed\_output\_fixed$parameters$gamma$samples), aes(x = Gamma)) +  
 geom\_density(fill = "#01454f", alpha = 0.3)



Posterior density of the gamma recovery/removal parameter, plotted using ggplot2

## Conclusions