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

##### Ciara Judge1,2,5, Timothy Vaughan3,4, Timothy Russell1,2, Sam Abbott1,2, Louis du Plessis3,4, Tanja Stadler3,4, \*Oliver Brady1,2, \*Sarah Hill5

1.  Department of Infectious Disease Epidemiology and Dynamics, Faculty of Epidemiology and Public Health, London School of Hygiene and Tropical Medicine, United Kingdom

2.  Centre for Mathematical Modelling of Infectious Diseases, London School of Hygiene and Tropical Medicine, United Kingdom

3.  Department of Biosystems Science and Engineering, ETH Zurich, Basel, Switzerland

4.  Swiss Institute of Bioinformatics, Lausanne, Switzerland

5.  Department of Pathobiology and Population Sciences, Royal Veterinary College, United Kingdom

\*[ciara.judge@lshtm.ac.uk](mailto:ciara.judge@lshtm.ac.uk); [sahill@rvc.ac.uk](mailto:sahill@rvc.ac.uk); [oliver.brady@lshtm.ac.uk](mailto:oliver.brady@lshtm.ac.uk)

## Introduction (for thesis)

In the preceding chapters of this thesis, I devised, validated and implemented a new model for inferring a range of outbreak characteristics (infection trajectories, trajectories, cumulative infection trajectories, etc) conditioned on phylogenetic tree and case incidence data. The model, ‘EpiFusion’ is packaged as a standalone Java executable software to be distributed for universal use. To improve utility, I also developed an R package ‘EpiFusionUtilities’ which facilitates convenient preparation of model data, parameters and priors in R and subsequent analysis of results. Further, a stable build of the EpiFusion model is included in the package, allowing the full workflow to be completed within an R session. These softwares comprise the EpiFusion Analysis Framework. For the purposes of reproducibility and open research, in this article I outline instructions on the implementation of the framework and provide a series of three vignettes where I demonstrate approaches to analysing simulated data. This article also details the addition of a new feature for incorporating phylogenetic uncertainty by providing a tree posterior samples from bayesian tree inference (from a program such as BEAST) as the phylogenetic tree data, in place of a fixed tree.

## 1. Abstract

The fields of epidemiology and viral phylodynamics share the ultimate goal of disease control, but concepts, methodologies and data employed by each differ in ways that confer complementary strengths and different areas of weakness. 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, including an example using a new feature incorporated since EpiFusion’s last description: the option to provide a phylogenetic tree posterior as the phylogenetic data input to the program. By outlining these use cases we aim to improve the usability and accessibility of our model, and promote workflow reproducibility and open research.

## 2. Introduction

Utilising computational tools for the implementation of mathematical models of infectious disease outbreak characteristics is an important aspect of public health research [1,2]. Modern advances in computation have facilitated the development of many such tools often distributed as packages or libraries in popular programming languages such as R, Python or Julia, or as standalone executable software [3,4,5,6]. Distributing and documenting epidemiological models and workflows is of vital importance for the purposes of reproducible research, and to enable efficient implementation of these models to answer research questions [7,8,9]. Previously we outlined EpiFusion, a novel method for modelling infectious disease outbreak characteristics conditioned on case incidence and phylogenetic trees using particle filtering, and validated its usage to infer infection trajectories and the effective reproduction number R(t) [10]. Here, we present the EpiFusion Analysis Framework, consisting of this EpiFusion model implemented as a Java command line tool, and the EpiFusionUtilities R package for data and output processing.

EpiFusion consists of a single process model, dual observation model particle filtering structure, where particles simulate outbreak trajectories and characteristics through time (process model) and are evaluated against phylodynamic and epidemiological data at resampling steps (observation model). EpiFusion uses case incidence and phylogenetic tree(s) as its data input, but can also be run with either data type alone (however, for the purposes of this article we focus on examples where we use both data types together). The force of infection over time is fit within the particle filter, and the recovery rate, case sampling rate, and genomic sequence sampling rate ( , and respectively) are fit via MCMC. Further information including model theory and validation are provided in [10].

The EpiFusion model is packaged as a Java [11] command line tool and takes eXtensible Markup Language (XML) files [12] as input. However, the recommended workflow for implementation includes data pre-processing, parameterisation, and eventual output parsing from within an R session, using the R package EpiFusionUtilities (<https://github.com/ciarajudge/EpiFusionUtilities>). It is also possible to call the EpiFusion executable from within R using the same package. Here we outline the EpiFusion Analysis Framework and demonstrate start-to-finish use cases for complete analysis of two outbreak datasets using EpiFusion and EpiFusionUtilities.

## 3. Methods

### Operation

EpiFusion is implemented as an 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 (<https://github.com/ciarajudge/EpiFusion/releases>). The source code for the latest development version is also available at this repository for users who wish to clone the repository and compile the program from source. EpiFusion can be called using its full file path, or 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 the R package devtools [13]:

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

The recommended EpiFusion workflow includes data processing and parameterisation in R using EpiFusionUtilities, 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

The EpiFusion Analysis Framework consists of three main steps: (i) Data Processing and Parameterisation (ii) Running EpiFusion and (iii) Parsing and Interpreting the output (*Figure 1*). All three steps can be carried out using EpiFusionUtilities, but it is also possible to manually assemble an XML input file and run EpiFusion from the command line using an executable jar file, which may be preferable in some circumstances (e.g. if using a remote system such as a high performance computing cluster).

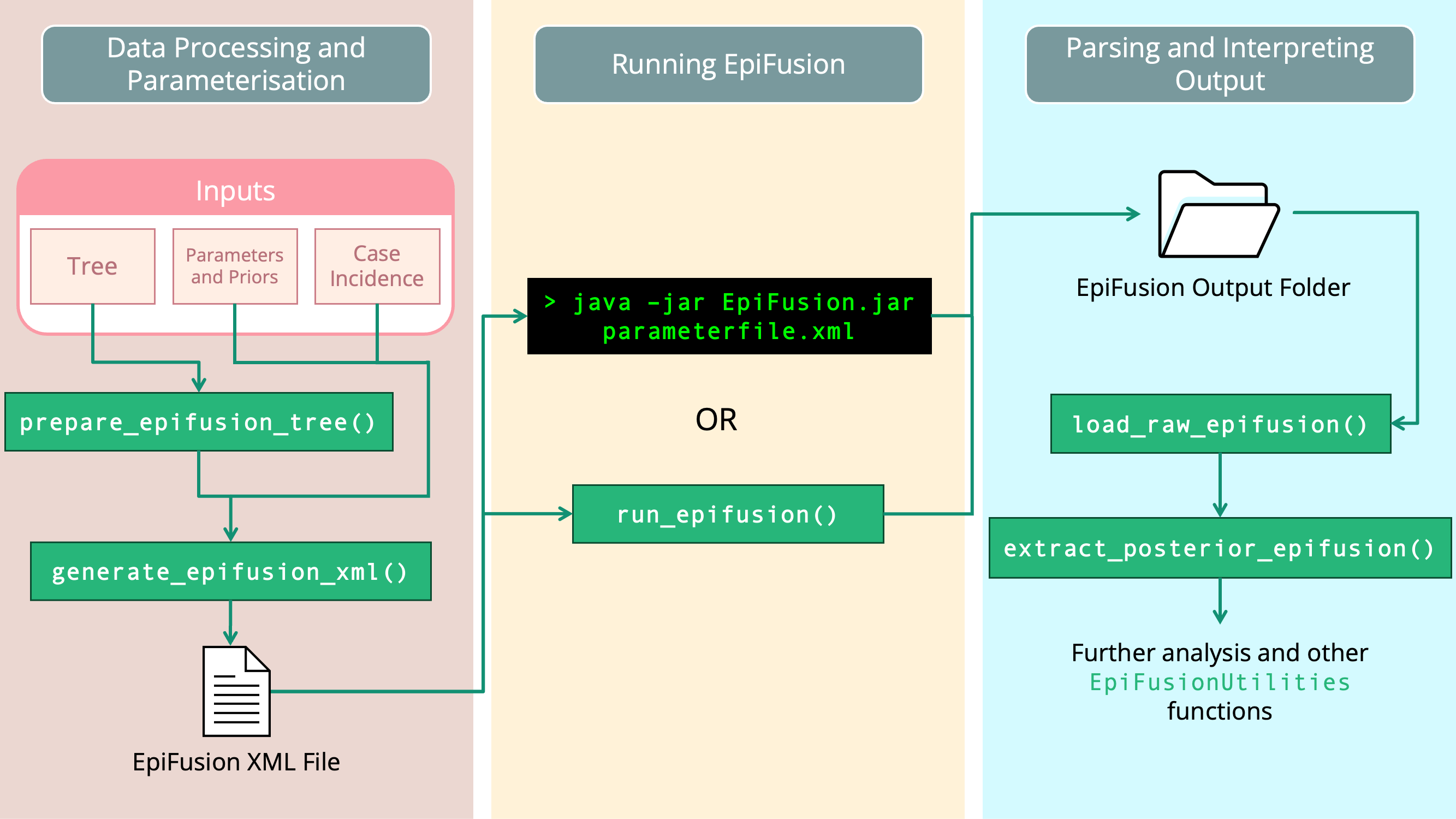


Figure 1 Recommended EpiFusion Framework workflow, using EpiFusionUtilities functions (green) to prepare data and parse results.

#### Data Processing and Parameterisation

##### 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 both 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 or trees. These can be supplied either directly within the XML document or by providing full 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 | Specifies 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 distribution specification for parameters to be fit via particle MCMC. A range of distribution options are available, including: Normal, Truncated Normal, Poisson, Uniform, Uniform Discrete, Beta and Fixed (if a parameter should be fixed to a specific value and not inferred via MCMC). |

##### Assembling parameter XML files

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

The first step of processing case incidence or tree data for EpiFusion input is to select an ‘index date’. The index date is the approximate date of origin of the outbreak (i.e. day ‘0’) and should be before any birth events in the tree or observed epidemiological cases. This date is provided to the processing functions to enable the case and incidence data to be rooted in numerical time units, all trajectory samples will assume that the outbreak originated with one individual becoming infected on the index date. All times in the EpiFusion input and output will be in relation to this date and measured in days.

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

If there is uncertainty in the date of origin of the outbreak we recommend arbitrarily setting the index date to earlier than the estimated date of origin. The resulting trajectories will likely demonstrate high uncertainty and low prevalence during the earliest days of the outbreak, until the point of the time series at which some data becomes available.

To prepare a tree or posterior set of trees for EpiFusion, pass an S3 phylo or multiPhylo object in R to the prepare\_epifusion\_tree() function (we recommend any standard phylogenetics R package to read in or generate these [14,15]). 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). This function adds node and leaf labels to the tree string that correspond to their time in days after the index date.

prepare\_epifusion\_tree(tree,  
 index\_date,  
 last\_sequence\_date,  
 "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 (included with the EpiFusionUtilities package) with the phylogenetic and/or case incidence data, and has default settings for 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 is 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 = logger\_information,  
 parameters = parameters\_to\_adjust,  
 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 (i.e. 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(). An installation of Java is still required.

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 daily trajectory of the force of infection over time sampled from the MCMC
* **trajectories:** .csv file where each row is a daily trajectory sampled from the MCMC of the number of individuals infected over time
* **params:** .txt file where each column is an MCMC parameter, and each row is an MCMC sample
* **likelihoods:** .txt file of the posterior likelihoods calculated at 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, to record which 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 functions to do this 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)

Below we demonstrate the implementation of this workflow using the EpiFusion executable and EpiFusionUtilities to analyse data from a small simulated outbreak.

## 4. Use Cases

### Description of the simulated datasets

In this article we demonstrate three analyses on two simulated outbreak datasets (*Figure 2, 3*). First we address a simple ‘baseline’ outbreak of length approximately three months with constant sampling consisting of a single epidemic peak. In ‘Full Framework Workflow’ we will model this outbreak using case incidence data in conjunction with a fixed time-scaled phylogenetic tree. Next, in ‘Phylogenetic Uncertainty’ we will model the same outbreak while incorporating phylogenetic uncertainty by using a tree posterior generated using a BEAST analysis of genomic samples simulated from the outbreak. Finally, in ‘Introducing Rate Changes’, we examine an outbreak with similar transmission dynamics but initial minimal sampling of cases and sequences followed by a sharp increase in the sampling rate, and demonstrate how to parameterise this in an EpiFusion model.

To generate the data, outbreak trajectories, and resulting weekly case incidence and a transmission tree of cases were simulated using ReMaster [16]. The transmission trees were downsampled to give a simulated phylogenetic tree of ‘sequenced samples’ from the outbreak, and for the baseline outbreak genomic sequences were simulated in R from this phylogenetic tree using the function simSeq() from the R package phangorn[15]. These sequences were used to generate a tree posterior using BEAST 2.7.3 [17] with a Birth Death Skyline model [18], under a strict clock and JC69 substitution model.

The date of origin of each outbreak was arbitrarily chosen as January 1st 2024 and the case data and tree leaves were labelled accordingly. 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 (<https://github.com/ciarajudge/EpiFusion_Vignettes>), where the code below is provided to fully replicate this use case example. The data is also provided directly as part of the EpiFusionUtilities package, and can be loaded directly into R using the functions baseline\_dataset and sampling\_dataset.

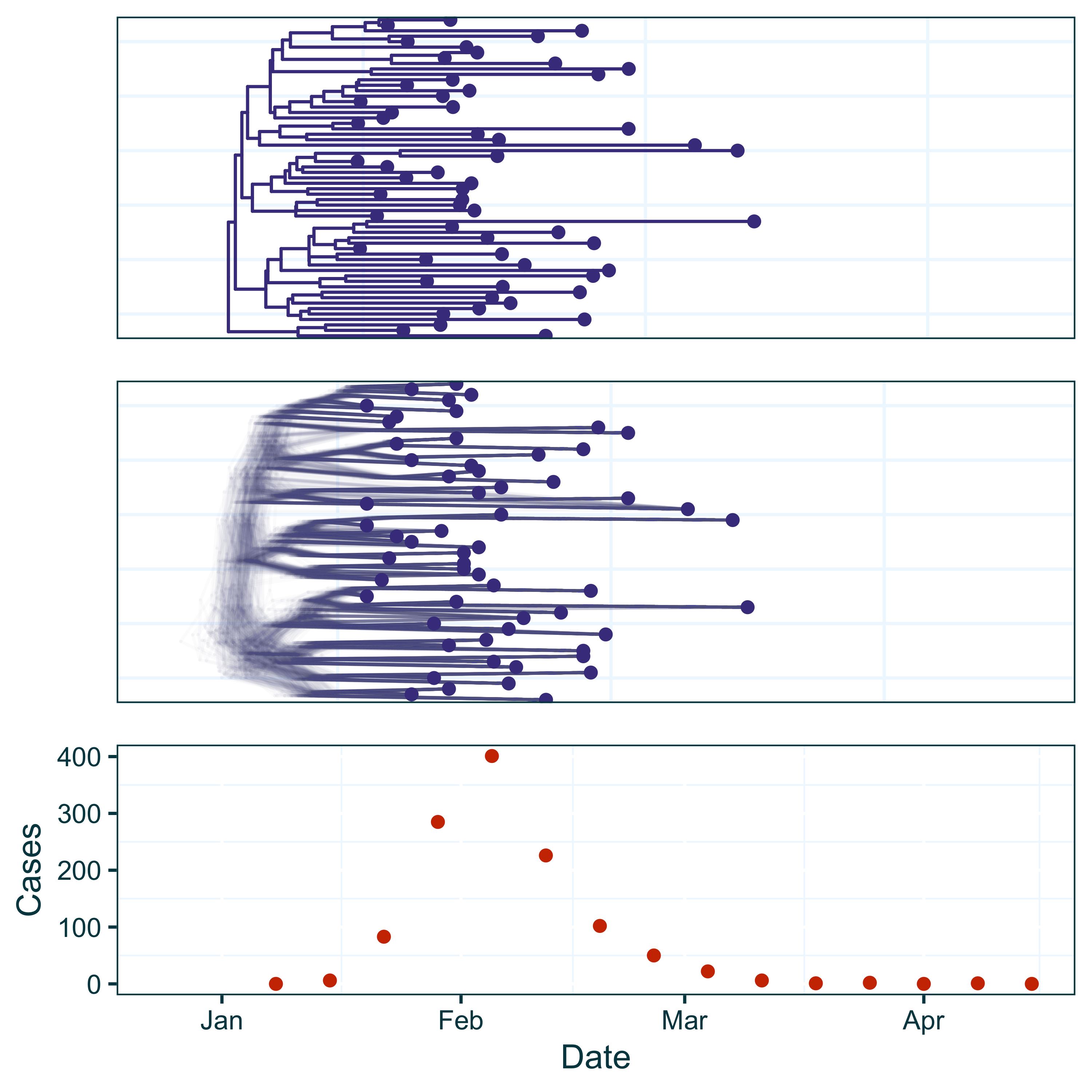


Figure 2 Data from the baseline outbreak simulated using ReMaster and BEAST 2.7.3. The dataset contains both a fixed phylogenetic tree (top) and tree posterior samples (middle), and weekly case incidence (orange).

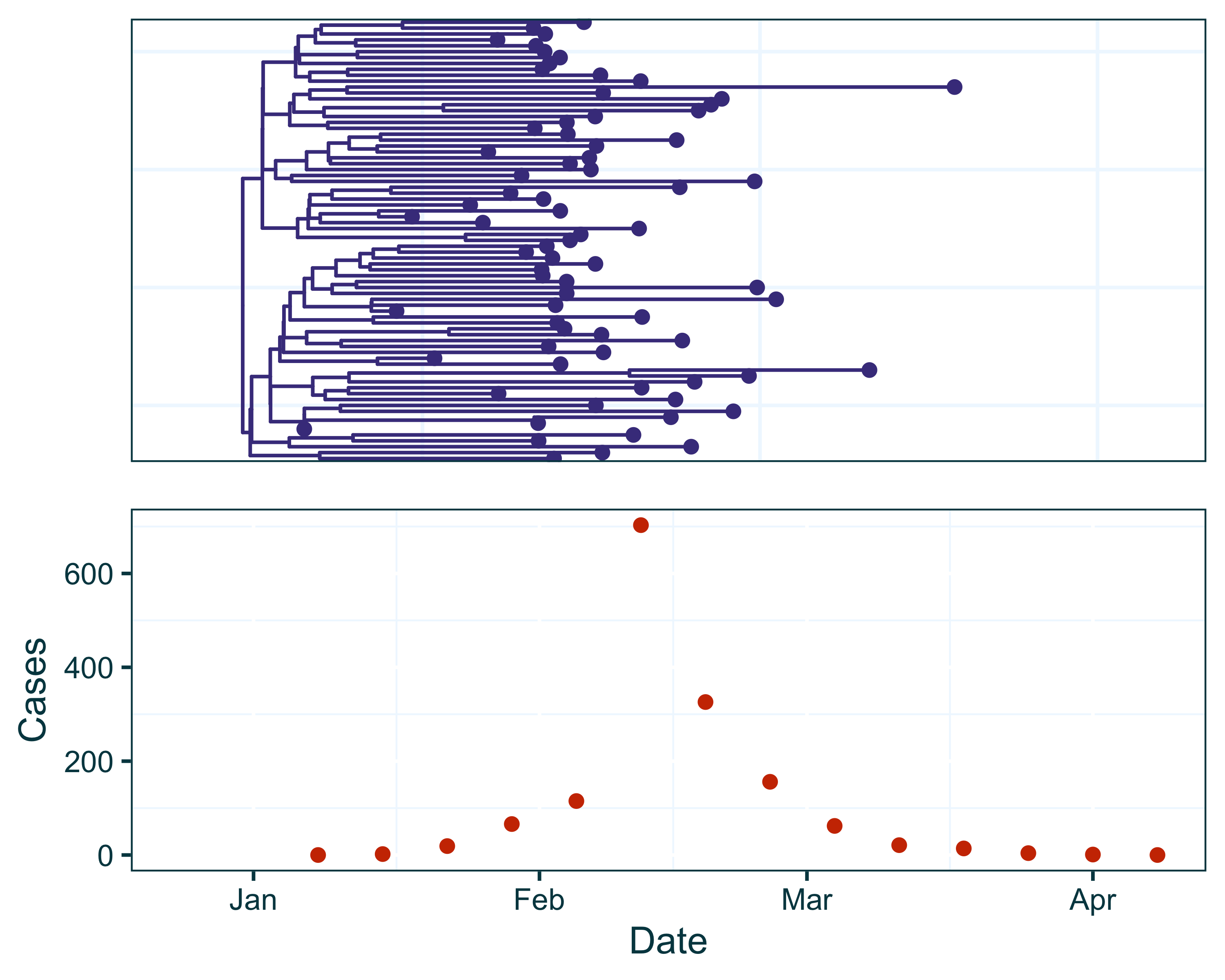


Figure 3 Data from outbreak with a step-change increase in sampling simulated using ReMaster. The dataset contains a fixed time-scaled phylogenetic tree and weekly case incidence. In this simulation, sampling of both epidemiological cases and genomic sequences was increased on February 5th.

### Full Framework Workflow

In this example, we will use the baseline dataset to show a full workflow using the EpiFusion Framework: (i) data preparation (ii) prior and parameter specification (iii) running EpiFusion (iv) parsing and plotting output.

#### Data Preparation

First we load and inspect the data for this example using the EpiFusionUtilities function baseline\_dataset(). This function loads a data frame with weekly case incidence (formatted with two columns, Cases and Date), a fixed time-scaled phylogenetic tree of samples, and samples from a tree posterior (with burn-in removed) from a BEAST analysis which we will use in a later section (Phylogenetic Uncertainty).

baseline\_dataset()  
  
print(baseline\_caseincidence[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(baseline\_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(baseline\_treeposterior)

## 200 phylogenetic trees

Next we set two date objects: the ‘index date’, or the earliest date from which we will model the outbreak origin date, and the date of sampling of the last observed sequence from the dataset. Whilst for this example we know (through the simulation process) that the outbreak origin was the 1st of January 2024, it can be good practice to set the index date to some time before the date that we suspect the outbreak began in the location represented by our case and phylogenetic data, to ensure the dynamics of full outbreak are captured.

index\_date <- as.Date("2023-12-26")  
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(baseline\_tree, index\_date, last\_sequence, "Data/Processed/baseline\_fixed\_tree.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>

We will generate an EpiFusion XML using the fixed tree we prepared with the prepare\_epifusion\_tree function and our loaded case incidence data. 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 that details how often we sample from the MCMC (every 10 MCMC steps):

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

To override this, 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/baseline\_fixed\_tree", logEvery = 5)

We will also slightly adjust the prior for initialBeta, or (force of infection at the beginning of the time series) from the default settings. As the default prior for is a truncated normal distribution with mean , by setting the initial value as we indicate that the initial is approximately between and .

priors <- list(initialBeta = list(stepchange = "false",  
 disttype = "Uniform",  
 min = 0.1,  
 max = 0.5))

In this example we are happy with the other parameters 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/baseline\_fixed\_tree.tree",  
 case\_incidence = baseline\_caseincidence,  
 index\_date = index\_date,  
 loggers = loggers,  
 priors = priors,  
 xml\_filepath = "Data/EpiFusion\_XMLs/baseline\_fixed\_tree\_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/baseline\_fixed\_tree\_inputfile.xml")

On conclusion of its analysis, EpiFusion saves a timings.txt file to the output folder with the total runtime in nanoseconds, which we examine and convert to minutes below:

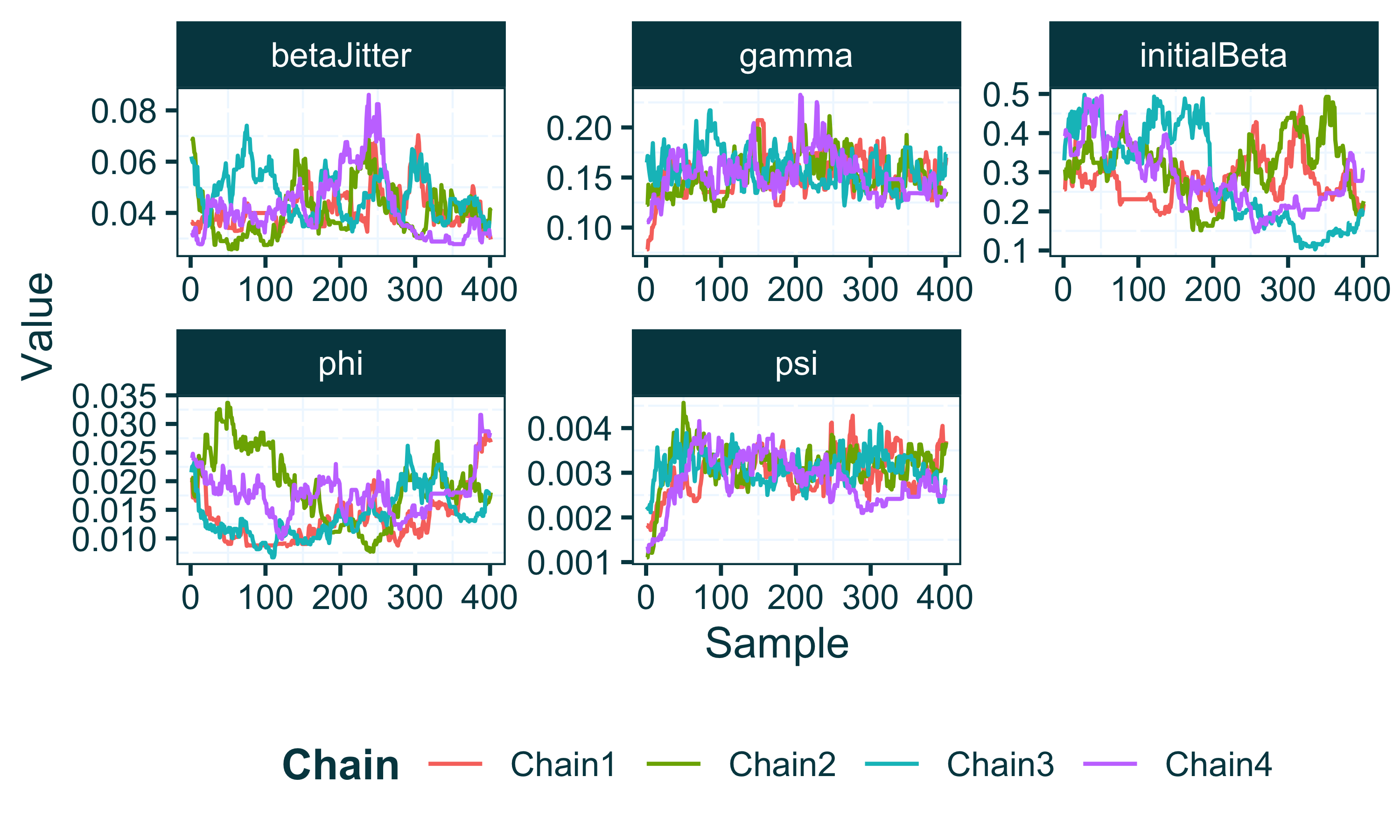
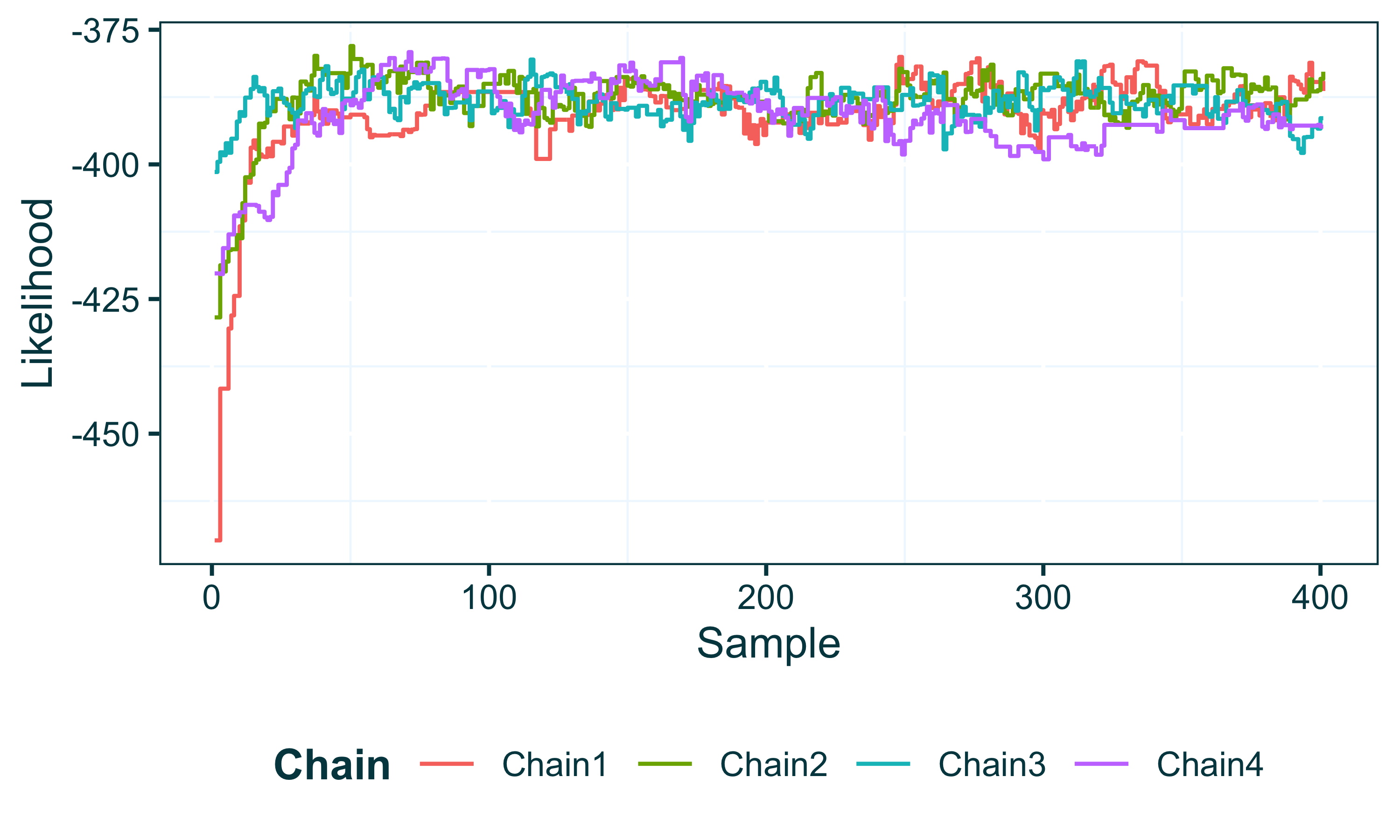
runtime <- suppressWarnings(read.table("Results/baseline\_fixed\_tree/timings.txt")[1,1]) / 6e10  
paste0("Runtime: ",runtime," minutes")

## [1] "Runtime: 7.76104354513333 minutes"

#### Parsing and plotting the output

First we will use the load\_raw\_epifusion function to import the full raw results. This function automatically produces plots (*Figure 4*) of the likelihood and parameter traces using the plot\_likelihood\_trace and plot\_parameter\_trace functions (these plots can be suppressed by passing the argument suppress = TRUE to the function). 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/baseline\_fixed\_tree/")



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. By default, the function returns means and Highest Posterior Density (HPD) intervals for the trajectories and parameters fitted by EpiFusion, however by specifying include\_samples = TRUE we also instruct the function to return the actual posterior samples (minus burn-in) for inspection. This greatly increases the memory used by the posterior output object in your R environment, so is recommended for initial inspection of your results but not for downstream tasks such as loading posteriors from many analyses for plotting.

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

## List of 5  
## $ infection\_trajectories:List of 3  
## ..$ mean\_infection\_trajectory : Named num [1:113] 0 1.2 1.46 1.69 1.96 ...  
## .. ..- attr(\*, "names")= chr [1:113] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ infection\_trajectory\_hpdintervals:List of 3  
## ..$ infection\_trajectory\_samples :'data.frame': 1448 obs. of 113 variables:  
## $ rt\_trajectories :List of 3  
## ..$ mean\_rt\_trajectory : Named num [1:113] 1.85 1.94 2.03 2.09 2.17 ...  
## .. ..- attr(\*, "names")= chr [1:113] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ rt\_trajectory\_hpdintervals:List of 3  
## ..$ rt\_trajectory\_samples :'data.frame': 1448 obs. of 113 variables:  
## $ parameters :List of 5  
## ..$ gamma :List of 3  
## ..$ psi :List of 3  
## ..$ phi :List of 3  
## ..$ betaJitter :List of 3  
## ..$ initialBeta:List of 3  
## $ fitted\_epi\_cases :List of 3  
## ..$ mean\_fitted\_epi\_cases : Named num [1:15] 0.131 11.057 82.289 298.367 412.858 ...  
## .. ..- 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': 1448 obs. of 15 variables:  
## $ cumulative\_infections :List of 3  
## ..$ mean\_cuminfection\_trajectory : Named num [1:113] 0 0.202 0.465 0.863 1.327 ...  
## .. ..- attr(\*, "names")= chr [1:113] "T\_0" "T\_1" "T\_2" "T\_3" ...  
## ..$ cuminfection\_trajectory\_hpdintervals:List of 3  
## ..$ cuminfection\_trajectory\_samples :'data.frame': 1448 obs. of 113 variables:

The extracted posterior object from the extract\_posterior\_epifusion function contains mean and HPD intervals of increasing width for infection, , 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 2023-12-26 0.000000 0 0 0  
## T\_1 2023-12-27 1.201657 1 2 1  
## T\_2 2023-12-28 1.464779 1 3 1  
## Upper88\_Infected Lower66\_Infected Upper66\_Infected Mean\_Rt Lower95\_Rt  
## T\_0 0 0 0 1.852666 0.9094750  
## T\_1 2 1 1 1.944617 0.8807066  
## T\_2 2 1 2 2.028745 0.7328984  
## Upper95\_Rt Lower88\_Rt Upper88\_Rt Lower66\_Rt Upper66\_Rt  
## T\_0 3.108418 0.9748502 2.820863 1.202323 2.340286  
## T\_1 3.182334 0.8807066 2.765279 1.190119 2.333551  
## T\_2 3.172166 1.1534527 3.061197 1.373219 2.515006  
## Mean\_CumulativeInfections Lower95\_CumulativeInfections  
## T\_0 0.0000000 0  
## T\_1 0.2016575 0  
## T\_2 0.4647790 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 1 0  
## Upper66\_CumulativeInfections  
## T\_0 0  
## T\_1 0  
## T\_2 1

It is possible use this table with ggplot functions to plot and inspect the inferred trajectories. However we also provide a function, plot\_trajectories that takes the trajectory table as input and automatically plots all three trajectory types (*Figure 5*).

plot\_trajectories(traj\_table)

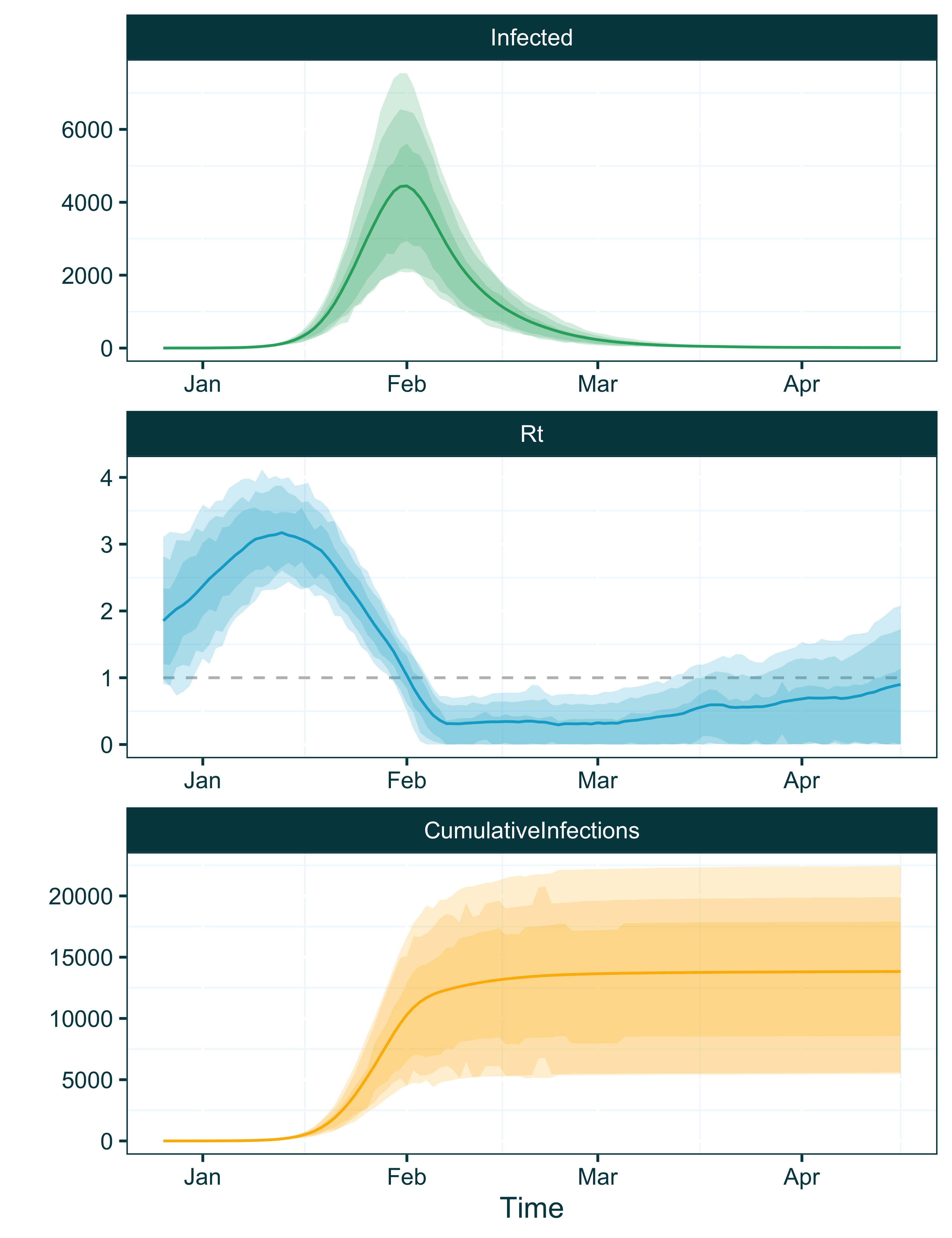


Figure 5 Infection, Rt and Cumulative Infection trajectories plotted by the EpiFusionUtilities function ‘plot\_trajectories’. For each trajectory type, the mean inferred trajectory is indicated by the coloured line, and the shaded regions of increasing darkness indicate 0.95, 0.88 and 0.66 HPD intervals.

The plot\_trajectories function also takes additional arguments to allow more customisation. For example, it is possible to provide a specific trajectory type to plot using the type argument, and specify bespoke plot colours using the plot\_colours argument. Here we will plot only the trajectories, in pink (*Figure 6*).

plot\_trajectories(traj\_table, type = "rt", plot\_colours = "pink")

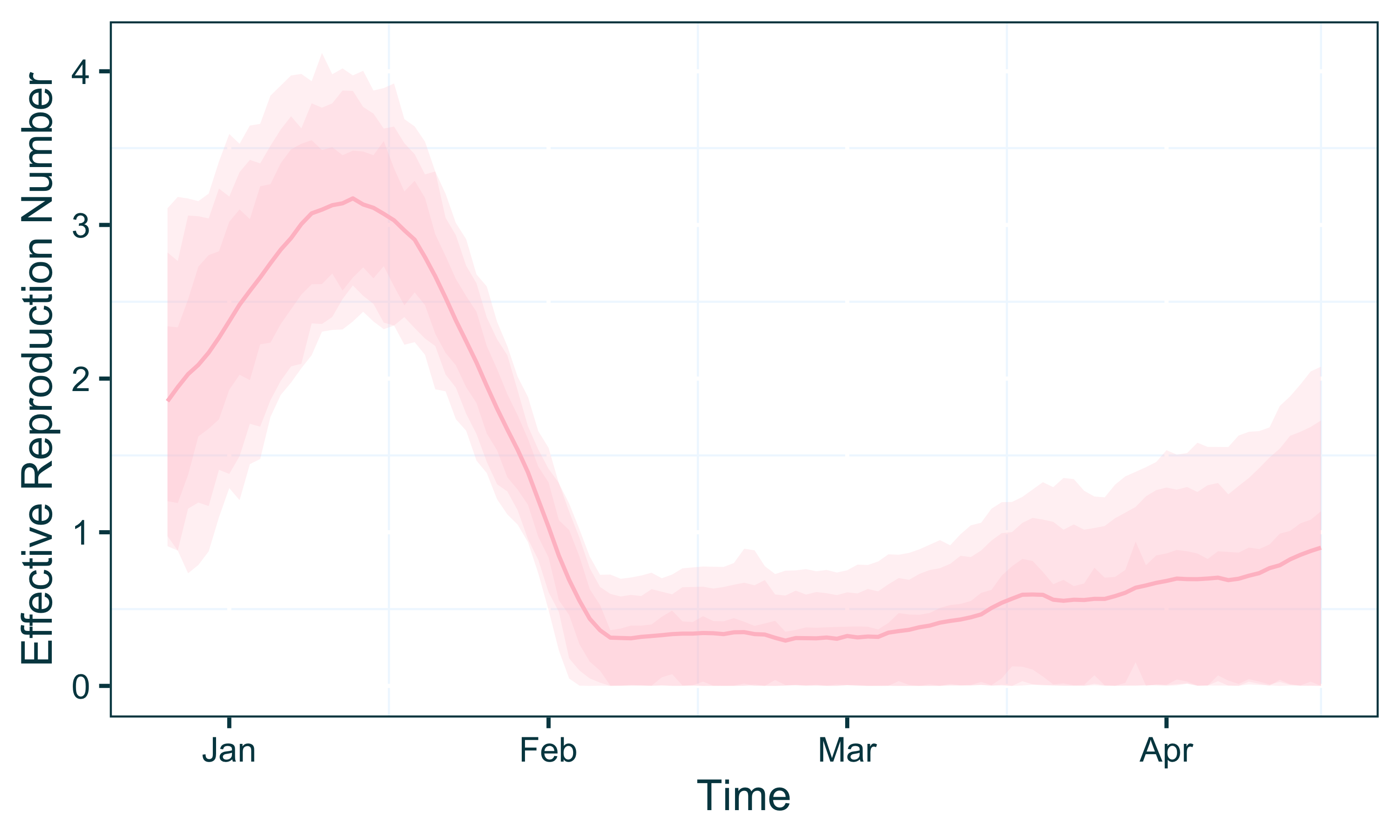


Figure 6 Inferred R(t) trajectories using a combined EpiFusion model and a fixed tree, plotted with the EpiFusionUtilities function ‘plot\_trajectories’. In this example, the ‘type’ argument was set to ‘rt’, so only the plot of the effective reproduction number trajectories is displayed.

As this was 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 (*Figure 7*).

epi\_data\_and\_fit\_table <- baseline\_caseincidence %>%  
 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\_and\_fit\_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()

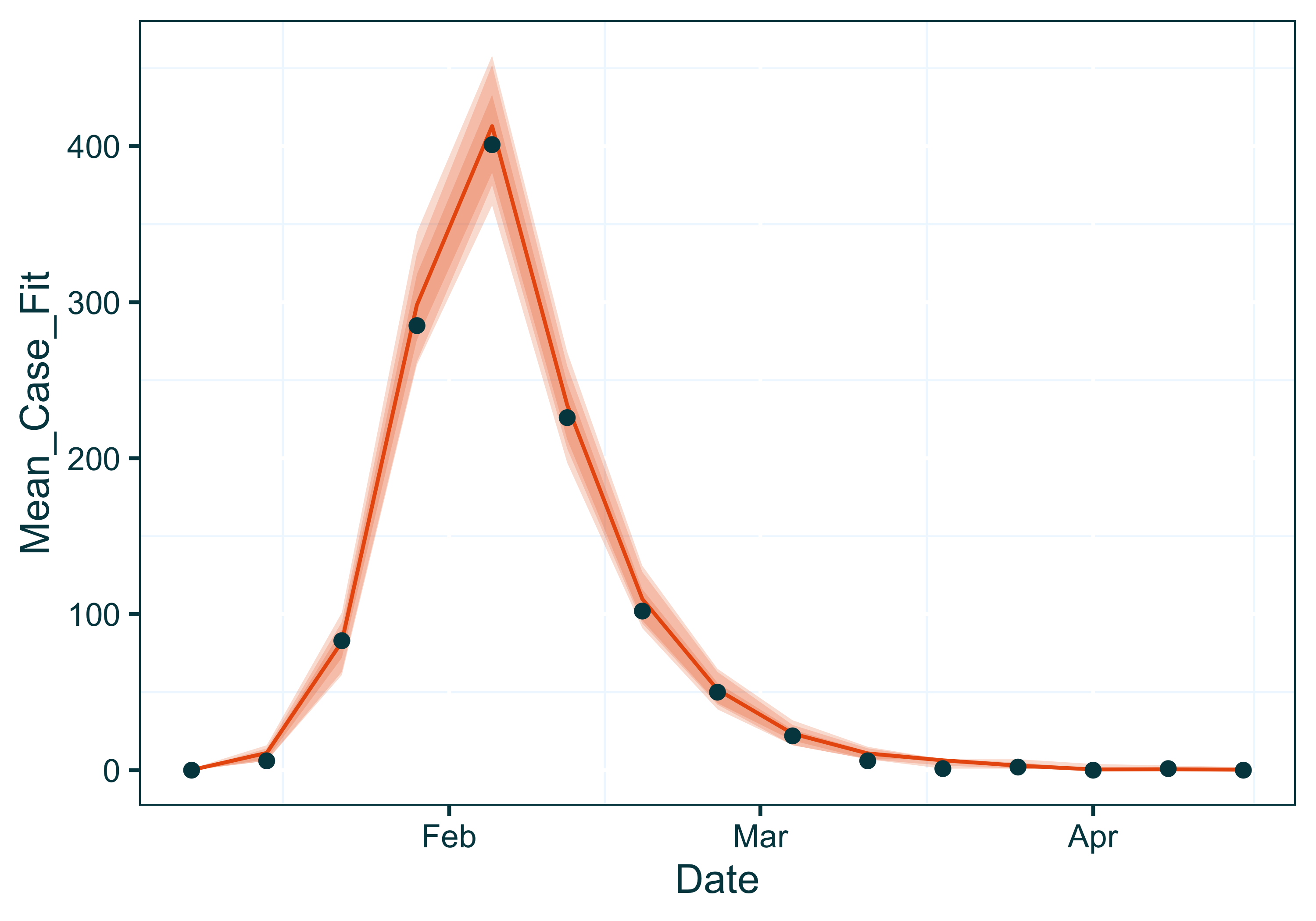


Figure 7 Fit of observed epidemiological cases to simulated cases by the EpiFusion model, plotted with ggplot2. Observed epidemiological cases are shown by the points, and the fitted case trajectory from EpiFusion is shown by the coloured line (mean) and shaded regions (HPD intervals).

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. If the gelman-rubin statistic is less than 1.015 this indicates MCMC convergence. For this example we used a minimal number of chains and steps (4 chains with 2000 MCMC steps) for time efficiency, but if the MCMC has not converged it may be necessary to run each chain for longer.

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

## [1] 1.00657

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

## [1] 249

We can also view the posterior density of a parameter by plotting the samples from the MCMC, which we can access from the posterior object due to setting include\_samples = TRUE when we extracted the posterior earlier using extract\_epifusion\_posterior (*Figure 8*).

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

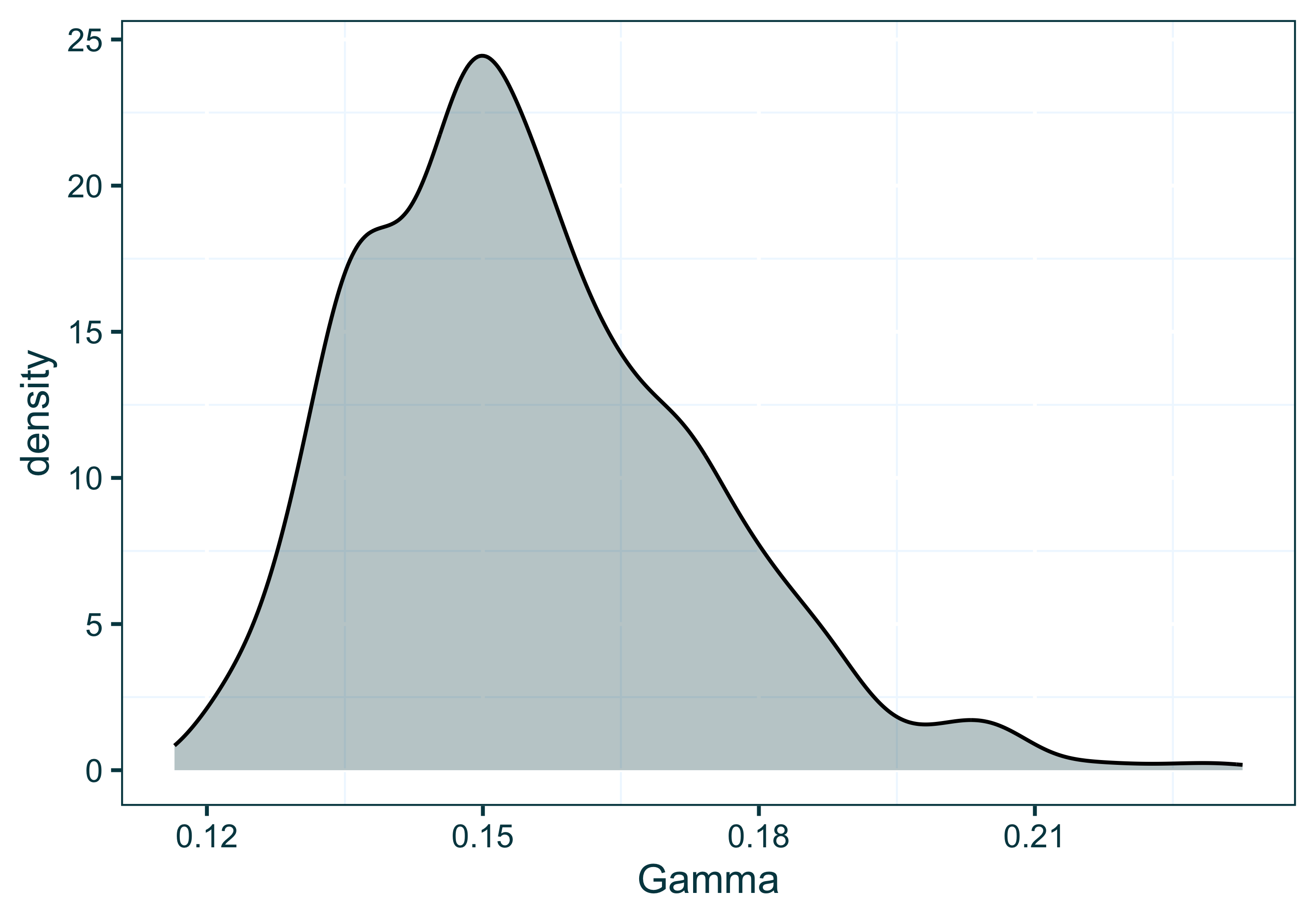


Figure 8 Posterior density of the gamma recovery/removal parameter, plotted using ggplot2

### Phylogenetic Uncertainty

In the previous example we modelled the baseline outbreak dataset using a fixed time-scaled phylogenetic tree as the phylodynamic data source. However, in a real outbreak setting there is often uncertainty about the true structure of this tree (phylogenetic uncertainty). Bayesian tree inference approaches such as BEAST [17,19] work to approximate the true tree by sampling trees from the posterior, often yielding thousands of plausible tree structures under the provided data and model. A single maximum clade credibility tree can be extracted from this ‘tree posterior’ and used in EpiFusion using fixed tree workflow specified above. However it may be preferable to be incorporate some of this phylogenetic uncertainty around tree structure into our EpiFusion model. Currently we achieve this by allowing the use of a tree posterior as the data input into EpiFusion, and sample a unique tree from the posterior for use with each MCMC chain. Below we demonstrate this approach by once more modelling the baseline outbreak dataset, but this time using the tree posterior as the phylodynamic data source.

#### Prepare Data and Parameters and Run EpiFusion

To prepare the tree posterior (which was already loaded into our environment when we used the baseline\_dataset() function) we once again use the prepare\_epifusion\_tree function. This function will recognise that a tree posterior has been passed, and will write the processed trees to a file without returning anything to your R session.

prepare\_epifusion\_tree(baseline\_treeposterior, index\_date, last\_sequence, "Data/Processed/baseline\_processed\_tree\_posterior.tree")

Next we will generate the XML file for the analysis using the tree posterior. We again specify adjustments to the loggers chunk, specifying our desired output folder name and how often to sample from the MCMC and print to console. We will also increase the number of MCMC chains to 50, which, in conjunction with passing a tree posterior to EpiFusion, will instruct the model to run 50 chains with 50 samples from the tree posterior. Resultingly, this analysis will take longer.

loggers <- list(fileBase = "Results/baseline\_tree\_posterior", logEvery = 5)  
parameters <- list(numChains = 5000)

Similarly to the fixed tree example, we will adjust some of the default priors. As before, we will set the initial force of infection to between and . We will also narrow the prior slightly, and adjust betaJitter, which is the amount of variation we expect in between days.

priors <- list(initialBeta = list(stepchange = "false",  
 disttype = "Uniform",  
 min = 0.1,   
 max = 0.5),  
 betaJitter = list(stepchange = "false",  
 disttype = "Uniform",  
 min = 0.005,   
 max = 0.05),  
 phi = list(stepchange = "false",  
 disttype = "TruncatedNormal",  
 lowerbound = 0.0,   
 mean = 0.02,  
 standarddev = 0.005))

Finally we generate an XML file using these parameters and priors for input into EpiFusion and run it:

generate\_epifusion\_XML(tree = "Data/Processed/baseline\_processed\_tree\_posterior.tree",  
 case\_incidence = baseline\_caseincidence,  
 index\_date = index\_date,  
 loggers = loggers,  
 parameters = parameters,  
 priors = priors,  
 xml\_filepath = "Data/EpiFusion\_XMLs/tree\_posterior\_inputfile.xml")  
  
run\_epifusion("Data/EpiFusion\_XMLs/tree\_posterior\_inputfile.xml")

#### Inspecting each chain

To examine the results of the model using the tree posterior we will again load the raw results with load\_raw\_epifusion. This time we will set suppress\_plots to true.

raw\_phylouncertainty <- load\_raw\_epifusion("Results/baseline\_tree\_posterior/", suppress\_plots = TRUE)

To examine the effect the inclusion of the tree posterior has on the analysis, we can use another EpiFusionUtilities function plot\_chainwise\_trajectories. This function operates similarly to the plot\_trajectories function, but separates the trajectories by chain for inspection. This allows us to see how the sampled tree, which differs between each chain, affects the inferred trajectories (*Figure 9*).

plot\_chainwise\_trajectories(raw\_phylouncertainty, 0.2)

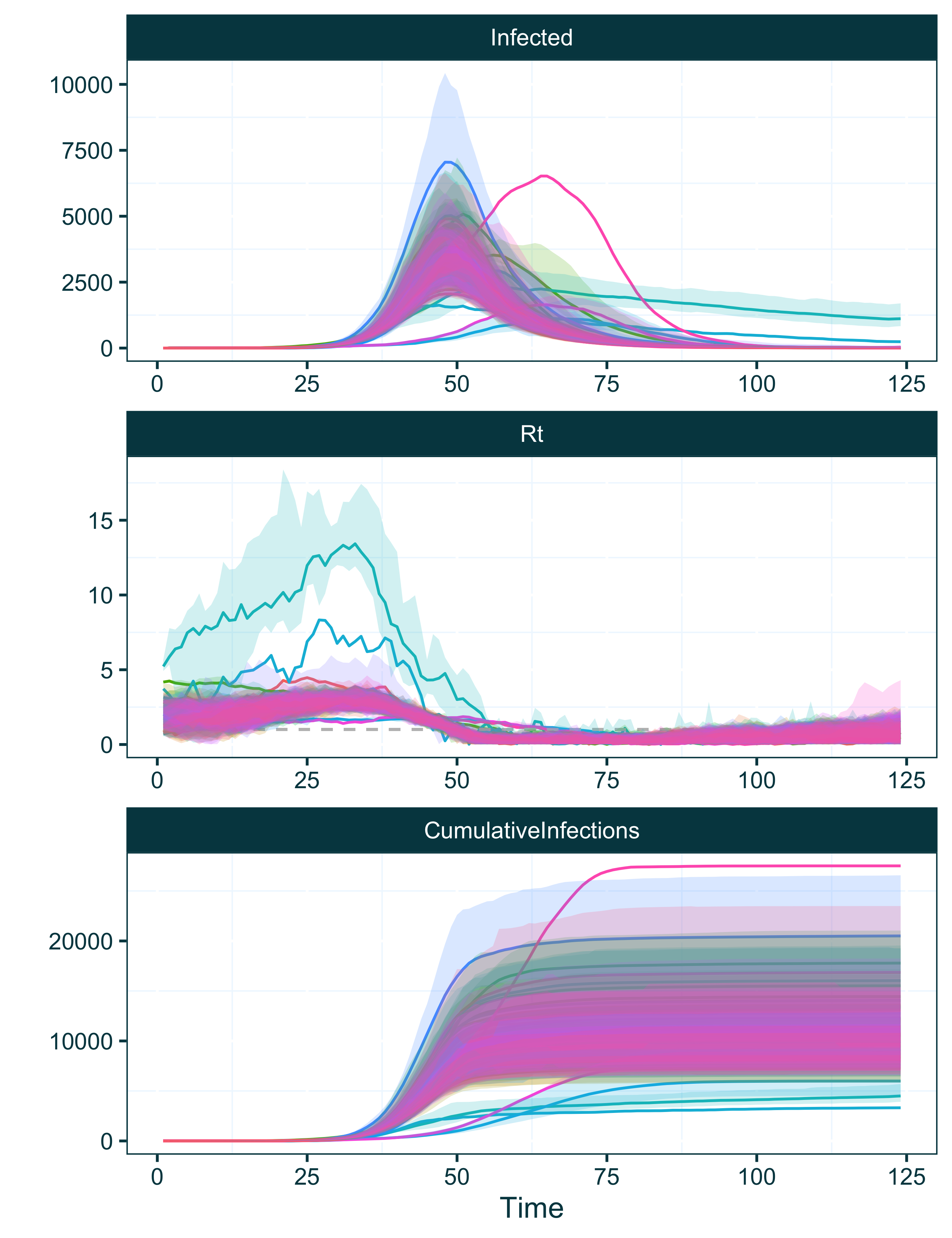


Figure 9 Inferred infection, and cumulative infection trajectories plotted using the ‘plot\_chainwise\_trajectories’ function of EpiFusionUtilities. The inferred trajectory means and HPD intervals of each chain are plotted and represented with a unique colour.

Here we can see that most of the chains converge on a similar set of trajectories to our fixed tree analysis, but some chains (and thus, some sampled trees) suggest other trajectory possibilities. When we extract the posterior from our raw output object, the chains will be combined and this uncertainty will be represented in our posterior estimates. To further understand this uncertainty we will extract the posterior sample using the extract\_posterior\_epifusion function and again create a trajectory table using the trajectory\_table function. Using this table, and our trajectory table from the fixed tree analysis, we can use ggplot2 to plot the trajectories from both analyses to demonstrate the effect of the phylogenetic uncertainty on the estimates (*Figure 10*).

posterior\_phylouncertainty <- extract\_posterior\_epifusion(raw\_phylouncertainty, 0.3)  
phylouncertainty\_trajtable <- trajectory\_table(posterior\_phylouncertainty, as.Date("2023-12-15")) %>%  
 mutate(Approach = "Tree Posterior")  
  
combined\_trajtable <- traj\_table %>%  
 mutate(Approach = "Fixed Tree") %>%  
 rbind(phylouncertainty\_trajtable)  
  
ggplot(combined\_trajtable, aes(x = Time, col = Approach, fill = Approach)) +  
 geom\_line(aes(y = Mean\_Infected)) +  
 geom\_ribbon(aes(ymin = Lower95\_Infected, ymax = Upper95\_Infected), col = NA, alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower88\_Infected, ymax = Upper88\_Infected), col = NA, alpha = 0.2) +  
 geom\_ribbon(aes(ymin = Lower66\_Infected, ymax = Upper66\_Infected), col = NA, alpha = 0.2) +  
 lshtm\_theme() +  
 facet\_wrap(~Approach, ncol = 1)

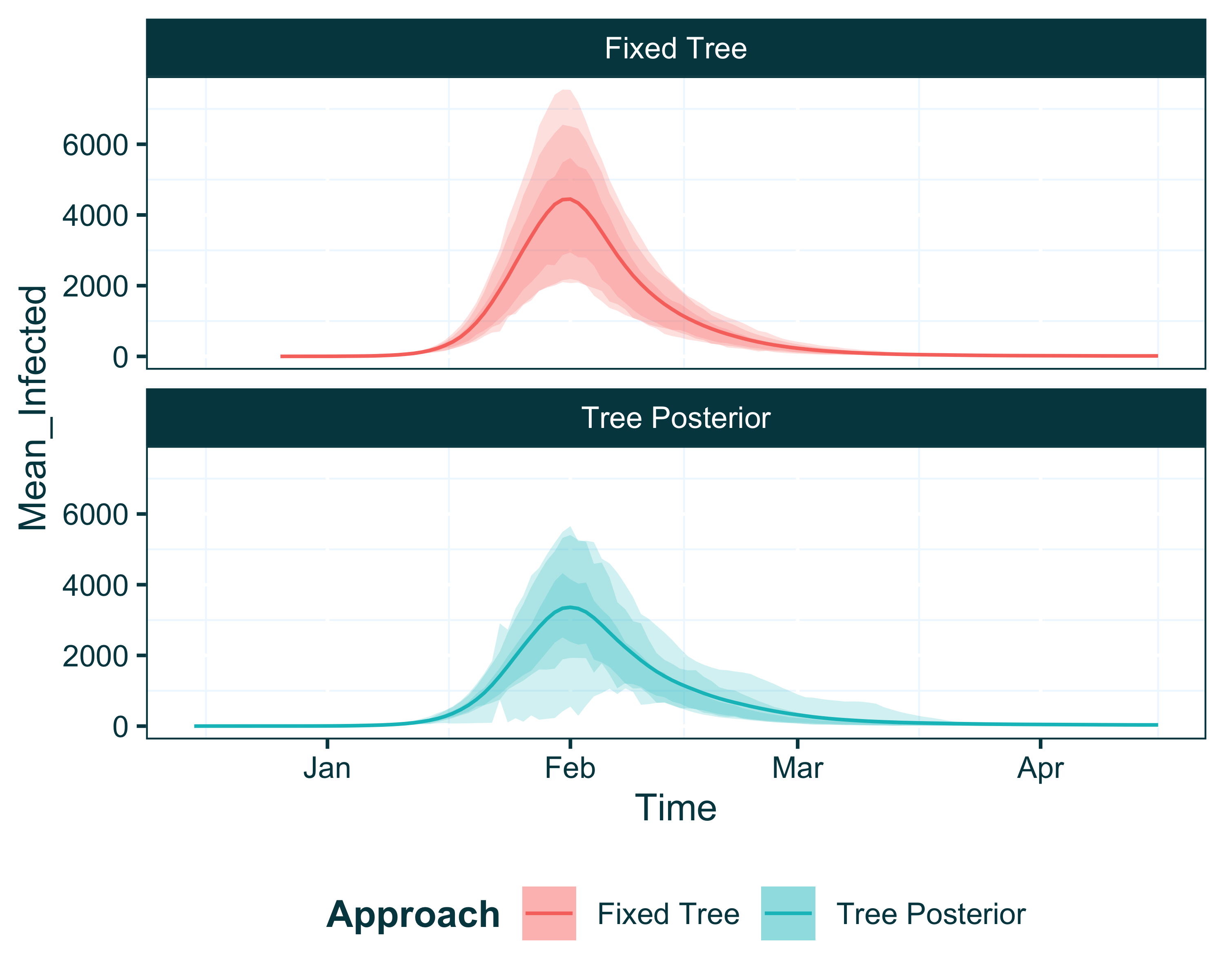


Figure 10 Inferred infection trajectories from EpiFusion analyses using a fixed tree (red) vs a tree posterior (blue).

The tree posterior approach is characterised by a widening of the HPD intervals around the mean fitted infection trajectory, due to the phylogenetic uncertainty.

### Introducing Rate Changes

While our previous examples with the baseline dataset describe a simple outbreak scenario with constant sampling throughout, real life scenarios are often more complicated as the rates that govern our model, , , and vary over time. is allowed to vary over time by default and is fit in the particle filter, but below we will address another plausible scenario where the case and genomic sequence sampling rates and are initially very low, followed by a sharp increase on February 5th (this is to simulate an example such as during the Brazilian Zika outbreak in 2015, where sampling sharply increased following the introduction of widespread PCR testing on December 8th [20]).

#### Data Preparation

We can load the data for this example using the EpiFusionUtilities function sampling\_dataset(). We will use the same index date for this analysis as previously, but for this dataset the last sequence in the tree was sampled on March 17th, so we will adjust the ‘last\_sequence’ date accordingly. As in our other examples, we prepare our tree data for EpiFusion using the prepare\_epifusion\_tree function.

sampling\_dataset()  
print(sampling\_caseincidence[1:5,])  
print(sampling\_tree)  
  
last\_sequence <- as.Date("2024-03-17")  
  
sampling\_fixed\_tree <- prepare\_epifusion\_tree(sampling\_tree, index\_date, last\_sequence, "Data/Processed/sampling\_fixed\_tree.tree")

#### Advanced Parameterisation: Time Variant Prior Distributions

In this example we wish to parameterise the step-increase in sampling on February 5th in our model. We will do this by setting a ‘time variant prior’ for case sampling rate phi when we generate the XML file, and using the ‘paired psi’ feature (*Supplementary Information Appendix 3*) to pair the genomic sampling rate psi to the case sampling rate.

Previously in the EpiFusion input files, the phi block in the prior section consisted of the following XML code:

<phi>  
 <stepchange>false</stepchange>  
 <disttype>TruncatedNormal</disttype>  
 <mean>0.02</mean>  
 <standarddev>0.01</standarddev>  
 <lowerbound>0.0</lowerbound>  
 </phi>

A phi parameter with a step change is adjusted to look like this:

<phi>  
 <stepchange>true</stepchange>  
 <changetime>  
 <x0>  
 <disttype>FixedParameter</disttype>  
 <value>35</value>  
 </x0>  
 </changetime>  
 <distribs>  
 <x0>  
 <disttype>TruncatedNormal</disttype>  
 <mean>0.002</mean>  
 <standarddev>0.0001</standarddev>  
 <lowerbound>0.0</lowerbound>  
 </x0>  
 <x1>  
 <disttype>TruncatedNormal</disttype>  
 <mean>0.025</mean>  
 <standarddev>0.005</standarddev>  
 <lowerbound>0.0</lowerbound>  
 </x1>  
 </distribs>  
 </phi>

The key differences here include the setting of the stepchange parameter to true, and the introduction of two new sub-nodes, changetimes and distribs, that contain the prior distribution details for the times of the rate changes in days from the index date (changetimes), and the rates themselves (distribs). For a rate with change points, there must be distributions in distribs and distributions in changetimes. These distributions are provided in tags with the format xn. While these adjustments can be made manually, it is also possible to parameterise this through the priors argument of generate\_epifusion\_xml using nested lists.

First we will make a list of the phi changetimes (in this example there is only one). In this scenario we ‘know’ the date of the step change in sampling - February 5th, 41 days after our index date - so we will provide it as a fixed parameter. However it is feasible to infer this change, if desired, by providing any discrete non-fixed prior distribution for this parameter.

phi\_changetimes <- list(x0 = list(disttype = "FixedParameter",  
 value = 41))

Next we will provide prior distributions for phi before and after the provided change time in the distribs.

phi\_distribs <- list(x0 = list(disttype = "TruncatedNormal",  
 mean = 0.005,  
 standarddev = 0.002,  
 lowerbound = 0.0),  
 x1 = list(disttype = "TruncatedNormal",  
 mean = 0.05,  
 standarddev = 0.02,  
 lowerbound = 0.0))

The list structure we introduce below using the changetimes and distribs we have created mirrors the structure of the XML chunk.

phi\_prior <- list(stepchange = "true",  
 changetime = phi\_changetimes,  
 distribs = phi\_distribs)

We can then feed this to the priors argument when we generate the XML file. We also will set pairedPsi to true in the parameters, and provide an empty pairedPsi in the priors. This specifies that psi is not to be fit by MCMC, and the genomic sampling rate psi is calculated as a proportion of the case sampling rate using the proportion of genomic sequences to cases in the data. Further information on this process is available in the Supplementary Information.

generate\_epifusion\_XML(tree = "Data/Processed/sampling\_fixed\_tree.tree",  
 case\_incidence = sampling\_caseincidence,  
 index\_date = index\_date,  
 loggers = list(fileBase = "Results/sampling\_step\_change", logEvery = 5),  
 parameters = list(pairedPsi = "true"),  
 priors = list(phi = phi\_prior,  
 pairedPsi = ""),  
 xml\_filepath = "Data/EpiFusion\_XMLs/sampling\_fixed\_tree\_inputfile.xml")  
  
run\_epifusion("Data/EpiFusion\_XMLs/sampling\_fixed\_tree\_inputfile.xml")

#### Parsing Results

To complete our analysis we will load our results using the load\_raw\_epifusion function once more, and inspect the parameter trace. Here we will suppress the automatically created plots, and plot the parameter trace ourselves using the plot\_parameter\_trace function (*Figure 10*).

raw\_sampling <- load\_raw\_epifusion("Results/sampling\_step\_change/", suppress\_plots = TRUE)  
  
plot\_parameter\_trace(raw\_sampling)



Figure 10 Parameter traces of MCMC parameters from an EpiFusion analysis with a case sampling rate () step change, plotted with the ‘plot\_parameter\_trace’ function. On the left, the parameter traces of the time-invariant parameters are plotted with line graphs where each chain is represented by a unique colour, the MCMC sample index is along the x-axis and the parameter value is on the y-axis. On the right, the values over the time series of the time-variant parameters are shown with a unique plot for each chain. The parameter value remains on the y-axis, but time in the time series is represented on the x-axis. The line colour denotes the MCMC sample index, with lighter blue samples representing samples from later in the MCMC chain.

Here the output from plot\_parameter\_trace looks slightly different to previous versions (e.g. Figure 4). The time-invariant parameters are plotted as before, with their traces from each chain plotted in unique colours on the left of the plot. On the right side of the plot, the function automatically recognises the presence of parameters that vary over time, and plots their piecewise constant values (y-axis) across time (x-axis) in step graphs. This allows the inferred value over time to be intuitively understood from the plot. The lines are coloured by their sample index on a continuous gradient, making visible the values to which the each chain has converged (light blue). Here we see that despite each chain initiating at different values, the initial and final sampling rates across each chain converge to approximately the same values. This is shown by the light blue (later MCMC samples) lines occurring at the same y-axis value in each chain trace plot for the phi parameter.

The parsing and plotting process for the remainder of the results from this analysis follows the same steps as the other vignettes included in this article.

## 5. Conclusions and Discussion

The EpiFusion Analysis Framework is a novel workflow for implementing the EpiFusion joint epidemiological and phylodynamic inference model using the java implementation of the model and the R package EpiFusionUtilities. This workflow is generalisable, utilising common R objects for data formatting (data.frame and phylo or multiPhylo objects) and parameterisation (list objects). We detail the full analysis workflow here, including a new feature introduced since EpiFusion’s first description: introduction of phylogenetic uncertainty by providing a tree posterior as data.

The current framework has been proven to be robust, but has some limitations. The joint inference model itself is written in java, with accompanying tools and functions available in R, a popular programming language among infectious disease epidemiologists. Traditionally R packages such as EpiFusionUtilities are distributed using the R package ecosystem CRAN. However, CRAN does not accept packages which contain binary executable code [21], so it would not have been possible to distribute an EpiFusion release with the package and enable running the model from within R.

The initial development of the model execution in java took place with the aim of eventual implementation of the model into the BEAST software ecosystem [17] as a post-hoc tool for processing the results of phylodynamic tree inference. This remains a likely next step, but an alternative approach may be to fully integrate the model into the EpiFusionUtilities R package using R/Rcpp, to allow more universal usage for all users including those without a extensive phylodynamic experience. This would also enable the package to be hosted on CRAN, as it would no longer be necessary to include a executable Java Archive (JAR) file.

The requirement to provide a user defined index date, or earliest possible date of outbreak origin, is a practical implementation compromise that may result in incorrect conclusions if the index date is not set early enough (i.e., if the index date is accidentally set after the true date of outbreak origin). Currently this can be overcome by arbitrarily setting the index date to a longer time period before the suspected origin of the outbreak, however future distributions of this framework will aim to allow inference without this truncation, or allow the index date to be inferred within the model.

The incorporation of phylogenetic uncertainty is now permitted in the program through allowing the use of a time-rooted phylogenetic tree posterior (from a software such as BEAST) to be used as data within the model. This new feature is an implementation rather than theoretical advancement: For each unique MCMC chain, a new tree is sampled from this posterior and used as the tree data in the model. The resulting posteriors can subsequently be examined and combined by the user with post-hoc EpiFusionUtilities functions. Incorporating phylogenetic uncertainty predictably led to increased uncertainty in the model estimates. We aim to investigate other approaches for incorporating phylogenetic uncertainty in the model in a more thorough manner.

To demonstrate the advanced parameterisation options of the framework, we addressed an outbreak with a step-increase in case and genomic sequence sampling rates. While this example featured a single change in the modelled rates, this infrastructure is very flexible and can be used to add significant complexity to the model according to the user requirements.

In the three vignettes described we provide examples of standard EpiFusion parameterisation, however there are a many advanced options available to users to customise their analysis. These include capabilities for composite (non-parametric) prior distributions (*Supplementary Information Appendix 4)*, multiple epidemiological observation model options (*Supplementary Information Appendix 2, ‘Model’*), multiple options for fitting (*Supplementary Information Appendix 5*), and buffer zones for rate step-changes (*Supplementary Information Appendix 6*).

In conclusion, this article aims to outline a reproducible framework for utilising our novel joint inference model using a functional R package and a binary executable file. We show how different parameterisations and options for the analysis can be implemented, including how to introduce phylogenetic uncertainty through the provided tree data, and time-variant prior distributions. We hope that clearly outlining a use case of the framework will facilitate its implementation by researchers to investigate hypotheses of public health importance in the future.

### Acknowledgements

The authors would like to thank Dr David Hodgson for his guidance on R package development and implementation, and Katie Tiley and Clara Brigitta for testing the package and vignettes.

## 6. Bibliography

[1] A. L. Jenner, R. A. Aogo, C. L. Davis, A. M. Smith, and M. Craig, “Leveraging Computational Modeling to Understand Infectious Diseases,” *Current Pathobiology Reports*, vol. 8, no. 4, p. 149, Sep. 2020, doi: [10.1007/s40139-020-00213-x](https://doi.org/10.1007/s40139-020-00213-x).

[2] D. Pokutnaya, B. Childers, A. E. Arcury-Quandt, H. Hochheiser, and W. G. V. Panhuis, “An implementation framework to improve the transparency and reproducibility of computational models of infectious diseases,” *PLOS Computational Biology*, vol. 19, no. 3, p. e1010856, Mar. 2023, doi: [10.1371/journal.pcbi.1010856](https://doi.org/10.1371/journal.pcbi.1010856).

[3] T. Jombart, M. Rolland, and H. Gruson, “CRAN task view: epidemiology.” Oct. 22, 2024. Available: <https://CRAN.R-project.org/view=Epidemiology>

[4] “Julia Epi.” Available: <https://github.com/JuliaEpi>

[5] D. J. Heslop, A. A. Chughtai, C. M. Bui, and C. R. MacIntyre, “Publicly available software tools for decision-makers during an emergent epidemicsystematic evaluation of utility and usability,” *Epidemics*, vol. 21, pp. 1–12, Dec. 2017, doi: [10.1016/j.epidem.2017.04.002](https://doi.org/10.1016/j.epidem.2017.04.002).

[6] “epirecipes: A cookbook of epidemiological models in R, Python, Julia (and more).” Available: <https://epirecipes.github.io/epicookbook/>

[7] A. S. Henderson, R. I. Hickson, M. Furlong, E. S. McBryde, and M. T. Meehan, “Reproducibility of COVID-era infectious disease models,” *Epidemics*, vol. 46, p. 100743, Mar. 2024, doi: [10.1016/j.epidem.2024.100743](https://doi.org/10.1016/j.epidem.2024.100743).

[8] K. T. L. Sy, L. F. White, and B. E. Nichols, “Reproducible Science Is Vital for a Stronger Evidence Base During the COVID-19 Pandemic,” *Geographical analysis*, Nov. 2021, doi: [10.1111/gean.12314](https://doi.org/10.1111/gean.12314).

[9] M. S. Jalali, C. DiGennaro, A. Guitar, K. Lew, and H. Rahmandad, “Evolution and reproducibility of simulation modeling in epidemiology and health policy over half a century,” *Epidemiologic Reviews*, vol. 43, no. 1, pp. 166–175, Dec. 2021, doi: [10.1093/epirev/mxab006](https://doi.org/10.1093/epirev/mxab006).

[10] C. Judge *et al.*, “EpiFusion: Joint inference of the effective reproduction number by integrating phylodynamic and epidemiological modelling with particle filtering,” *PLOS Computational Biology*, vol. 20, no. 11, p. e1012528, Nov. 2024, doi: [10.1371/journal.pcbi.1012528](https://doi.org/10.1371/journal.pcbi.1012528).

[11] “Java programming language.” Available: <https://docs.oracle.com/javase/8/docs/technotes/guides/language/>

[12] M. Lupp, “Extensible Markup Language,” S. Shekhar and H. Xiong, Eds., Boston, MA: Springer US, 2008, pp. 307–307. doi: [10.1007/978-0-387-35973-1\_400](https://doi.org/10.1007/978-0-387-35973-1_400).

[13] H. Wickham, J. Hester, W. Chang, J. Bryan, and RStudio, *Devtools: Tools to make developing r packages easier*. 2022. Available: <https://cran.r-project.org/web/packages/devtools/index.html>

[14] E. Paradis and K. Schliep, “Ape 5.0: An environment for modern phylogenetics and evolutionary analyses in r,” *Bioinformatics*, vol. 35, no. 3, pp. 526–528, Feb. 2019, doi: [10.1093/bioinformatics/bty633](https://doi.org/10.1093/bioinformatics/bty633).

[15] K. P. Schliep, “Phangorn: Phylogenetic analysis in r,” *Bioinformatics*, vol. 27, no. 4, 2011, doi: [10.1093/bioinformatics/btq706](https://doi.org/10.1093/bioinformatics/btq706).

[16] T. G. Vaughan, “ReMASTER: Improved phylodynamic simulation for BEAST 2.7,” *bioRxiv*, pp. 2023.10.09.561485–2023.10.09.561485, Oct. 2023, doi: [10.1101/2023.10.09.561485](https://doi.org/10.1101/2023.10.09.561485).

[17] R. Bouckaert *et al.*, “BEAST 2.5: An advanced software platform for bayesian evolutionary analysis,” *PLOS Computational Biology*, vol. 15, no. 4, pp. e1006650–e1006650, 2019, doi: [10.1371/JOURNAL.PCBI.1006650](https://doi.org/10.1371/JOURNAL.PCBI.1006650).

[18] T. Stadler, D. Kühnert, S. Bonhoeffer, and A. J. Drummond, “Birth-death skyline plot reveals temporal changes of epidemic spread in HIV and hepatitis c virus (HCV),” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 1, pp. 228–233, Jan. 2013, doi: [10.1073/PNAS.1207965110/SUPPL\_FILE/SAPP.PDF](https://doi.org/10.1073/PNAS.1207965110/SUPPL_FILE/SAPP.PDF).

[19] A. J. Drummond and A. Rambaut, “BEAST: Bayesian evolutionary analysis by sampling trees,” *BMC Evolutionary Biology*, vol. 7, no. 1, p. 214, Nov. 2007, doi: [10.1186/1471-2148-7-214](https://doi.org/10.1186/1471-2148-7-214).

[20] O. J. Brady *et al.*, “The association between zika virus infection and microcephaly in brazil 20152017: An observational analysis of over 4 million births,” *PLOS Medicine*, vol. 16, no. 3, pp. e1002755–e1002755, Mar. 2019, doi: [10.1371/JOURNAL.PMED.1002755](https://doi.org/10.1371/JOURNAL.PMED.1002755).

[21] “CRAN repository policy.” Available: <https://cran.r-project.org/web/packages/policies.html>