EpiFusion Tutorial

2023-12-20

Introduction

This R Markdown document will guide you through the process of parsing and interpreting the output of the EpiFusion tutorial. We will use a selection of functions that can also be found in the **EpiFusion_utilities.R** script on the GitHub repository.

The first step of the tutorial is to point the document to the filepath of the output folder made by EpiFusion. We will also source the `EpiFusion_utilities.R` script to load the necessary functions to look through this output. The current filepath assumes the folder is in your working directory; you may need to change this depending on where your output folder was generated.

```
outputfolder <- "intro-example/"
source("EpiFusion_utilities.R")
```

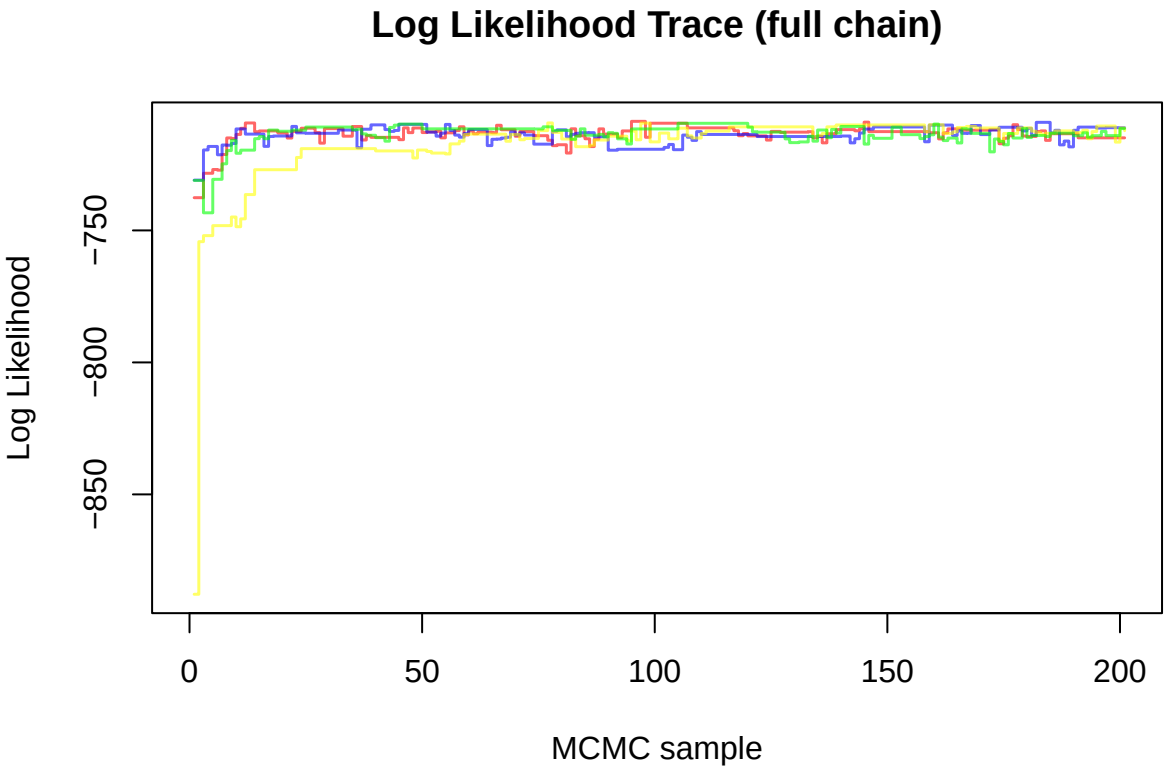
```
## Loading required package: pacman
```

```
## [1] "Loading EpiFusion functions."
```

NOTE: the first time you run this, the output might also show the installation of any required R packages you don't already have installed.

Checking convergence

The first thing we'll do is take a look at the trace plots of the chain likelihood, to check that the model has run nicely. For this we can use the `plotlikelihoodtrace` function, which takes the output folder filepath as its input.



Have a look at the likelihood trace for each chain run. They should start in different places but eventually converge to roughly the same value. Keep an eye out for chains getting stuck (staying at the same value for a long time). This does happen occassionally, and we are working on introducing adaptive Metropolis Hastings MCMC sampling to EpiFusion to fix this. Once you have had a look at the trace plot you can choose what proportion of each chain to discard as burn-in. EpiFusion models tend to converge relatively quickly, so the default in this document will be 10%. However, you can edit this as needed.

```
burn_in <- 0.1
```

Examining infection trajectory posteriors

Trajectories table

Next we'll examine the infection trajectories inferred by your EpiFusion model. First let's take a look at the raw values by loading the trajectories to our environment with `loadtrajectoriesminusburnin`, which takes the `outputfolder` and `burn_in` as arguments. This will return a table, with a column for every day of the analysis and a row for every sampled trajectory. Below we load the table and look at the first sample.

```
trajectories <- loadtrajectoriesminusburnin(outputfolder, burn_in)
unlist(trajectories[1,])
```

```
##   T_0   T_1   T_2   T_3   T_4   T_5   T_6   T_7   T_8   T_9  T_10  T_11  T_12
##     1     1     1     4     6     7     7    10    15    21    25    30    35
##  T_13  T_14  T_15  T_16  T_17  T_18  T_19  T_20  T_21  T_22  T_23  T_24  T_25
##    40    47    53    65    67    73    83    89   113   126   146   168   205
##  T_26  T_27  T_28  T_29  T_30  T_31  T_32  T_33  T_34  T_35  T_36  T_37  T_38
##   227   271   309   350   396   452   515   557   634   705   772   833   894
```

```
##  T_39  T_40  T_41  T_42  T_43  T_44  T_45  T_46  T_47  T_48  T_49  T_50  T_51
##   987  1066  1195  1294  1437  1611  1796  1987  2261  2589  2900  3210  3509
##  T_52  T_53  T_54  T_55  T_56  T_57  T_58  T_59  T_60  T_61  T_62  T_63  T_64
##  3765  3906  4063  4159  4154  4182  4167  4146  4060  3946  3781  3605  3470
##  T_65  T_66  T_67  T_68  T_69  T_70  T_71  T_72  T_73  T_74  T_75  T_76  T_77
##  3305  3174  3000  2842  2726  2560  2453  2289  2092  1886  1685  1552  1440
##  T_78  T_79  T_80  T_81  T_82  T_83  T_84  T_85  T_86  T_87  T_88  T_89  T_90
##  1343  1225  1109  1001   910   832   757   690   632   575   537   514   474
##  T_91  T_92  T_93  T_94  T_95  T_96  T_97  T_98  T_99 T_100 T_101 T_102 T_103
##   429   395   372   333   312   293   266   241   224   209   182   161   152
## T_104 T_105 T_106 T_107 T_108 T_109 T_110 T_111 T_112 T_113 T_114 T_115 T_116
##   145   136   125   117   106   102    95    88    80    75    65    61    50
## T_117 T_118 T_119 T_120 T_121 T_122 T_123 T_124 T_125 T_126 T_127 T_128 T_129
##    44    41    39    35    32    35    35    39    36    33    28    28    25
## T_130 T_131 T_132 T_133 T_134 T_135 T_136 T_137 T_138 T_139 T_140
##    25    22    19    18    17    17    15    15    15    15    15
```

You can also load the trajectories inferred by each chain separately. This can be useful if you think you might have convergence issues.

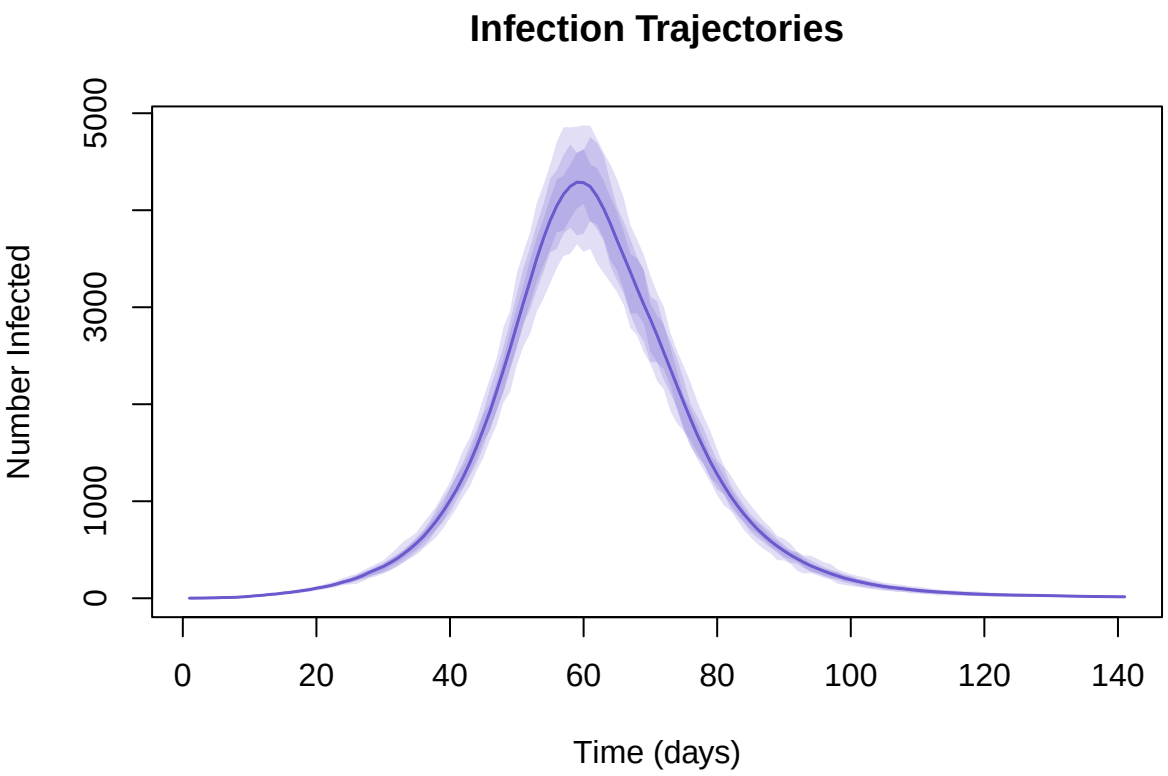
```
trajectories_separate <- loadtrajectoriesminusburninseparate(outputfolder, burn_in)
```

This function returns a list of length (N) of trajectory tables, where N = the number of chains.

Trajectories plot

It can be difficult to get a sense for what the trajectories are really saying in table form, so let's plot them using `plottrajectoriesfromtable`. We can supply our desired colour as the second argument to the function. There is also a function, `plottrajectoryposteriors`, which can plot the trajectories directly from the file, but we won't use that here. If you do want to us it, it takes `outputfolder`, `burn_in` and `colour` as arguments.

```
plottrajectoriesfromtable(trajectories, 'slateblue')
```



This function plots the mean sampled trajectory as a coloured line, with 95%, 80% and 66% HPDs as shaded regions of increasing darkness.

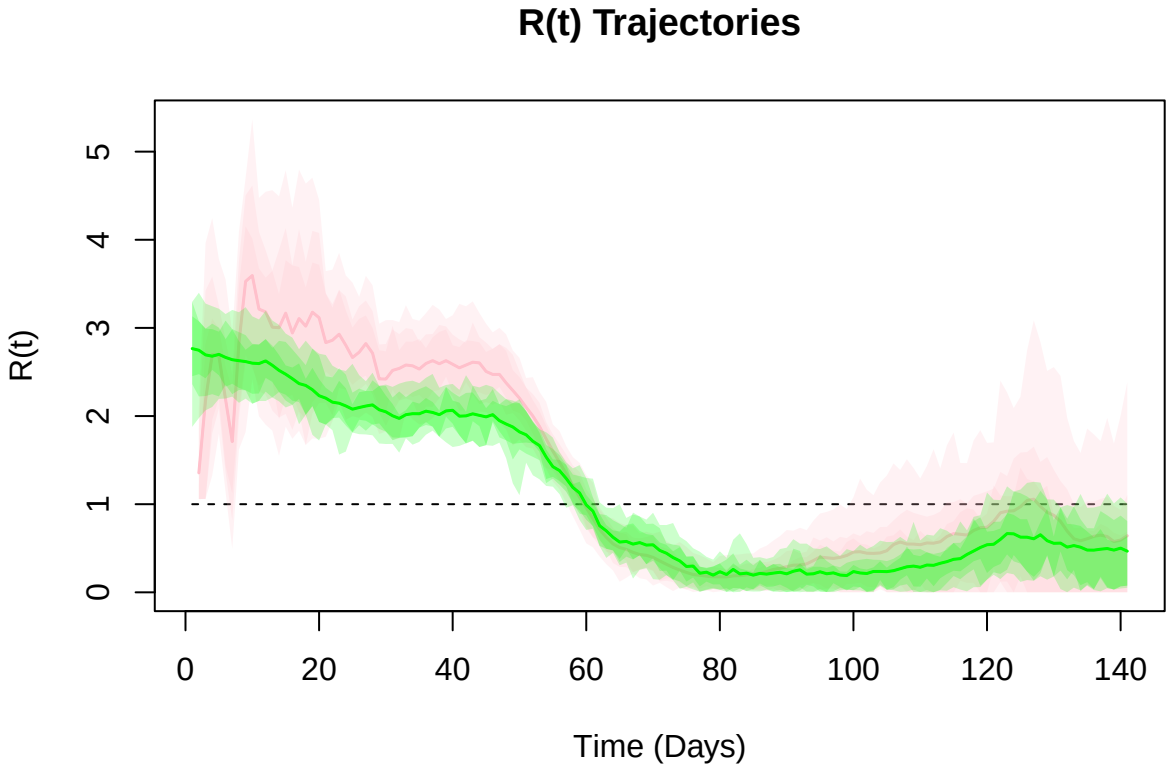
Examining R(t) trajectory posteriors

Now lets take a look at the R(t) estimates from the model. With EpiFusion it is actually possible to get R(t) through two methods. We'll examine both here.

Renewal equation or Beta/Gamma?

The EpiFusion program stores daily infection events in the trajectories that it models, and uses this along with a user specified probability mass function of the generation time in a renewal equation to calculate an R(t) trajectory for each MCMC sample. These are stored in the `rt_chainN.csv` files of the EpiFusion output. However, EpiFusion also samples trajectories of the force of infection beta over time (`betas_chainN.csv`). Together with the posterior samples for the recovery/removal parameter gamma, you can also get R(t) estimates from dividing beta(t)/gamma(t) for each MCMC sample. Lets plot both options together to compare:

```
plotrtposteriors_renewal(outputfolder, burn_in, 'pink')
addrtposteriors_betagamma(outputfolder, burn_in, 'green')
```



Here we've plotted the renewal R(t) posteriors with `plotrtposteriors_renewal` and added a layer of the beta/gamma R(t) posteriors with `addrtposteriors_betagamma`. There is also the inverse of each of these functions if you wanted to reverse the order; i.e. `addrtposteriors_renewal` and `plotrtposteriors_betagamma`. These R(t) trajectories should be in agreement, however the renewal equation version can be a little unstable when prevalence is low (at the start and end of this example). You can choose whichever option meets your needs.

Loading R(t) tables

You can load the R(t) in table form also, as shown below. The first value of the renewal equation trajectories will always be NaN due to the edge limitations of this method.

```
#Load renewal equation trajectories
renewal_rts <- loadrtsminusburnin(outputfolder, burn_in)
#Load beta/gamma trajectories
betagamma_rts <- loadbetagammartminusburnin(outputfolder, burn_in)
```

Similarly to the infection trajectories, there are also functions for plotting R(t) posteriors (or adding layers) directly from the tables. A list of all available functions is on the GitHub wiki, we recommend going through it to see what's available.

MCMC Parameter Posteriors

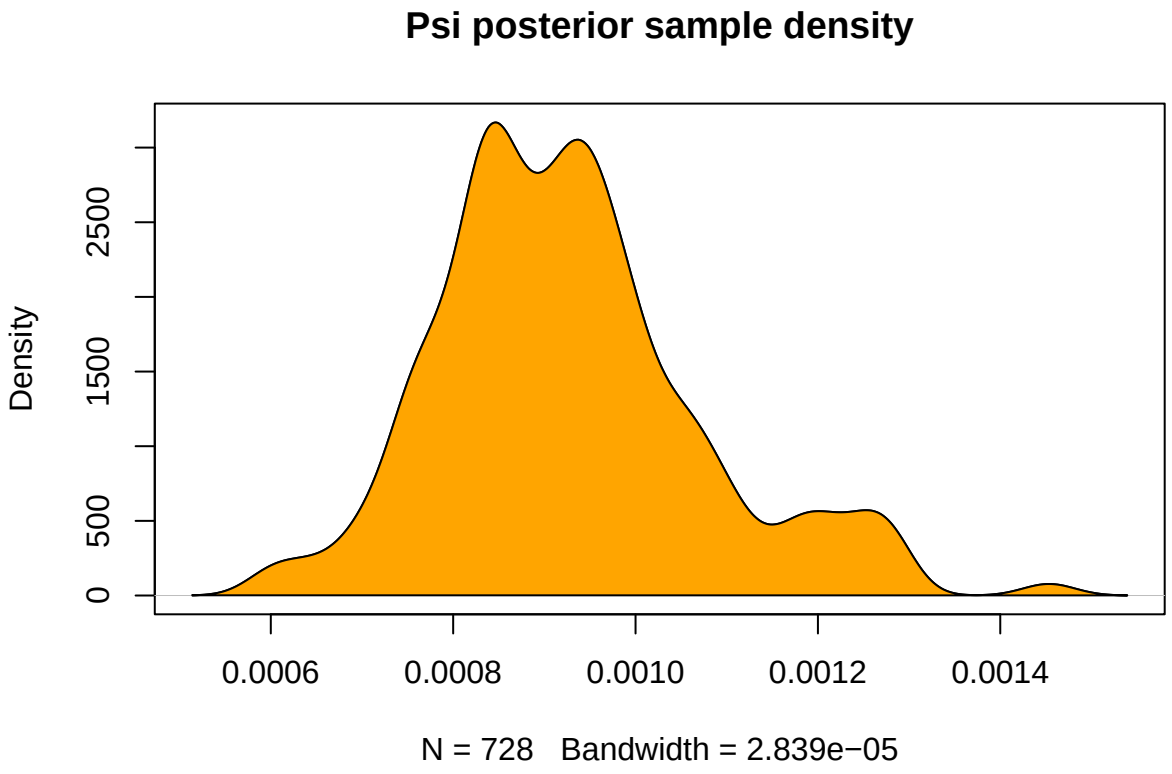
Another set of files you'll find in the EpiFusion are the `params_chainN.txt` files. These contain a row for each MCMC sample, and a column for each MCMC parameter. You can load them using the `loadparamsminusburnin` function. The last column of the table will always contain NAs, this is just an artefact of how EpiFusion saves the samples.

```
paramposteriors <- loadparamsminusburnin(outputfolder, burn_in)
head(paramposteriors)
```

```
##        gamma          psi        phi initialBeta betaJitter  X
## 20 0.1015715 0.0008628258 0.01966505   0.2898188 0.01303787 NA
## 21 0.1043629 0.0007787591 0.01939411   0.2795766 0.01628408 NA
## 22 0.1144166 0.0008406108 0.01896374   0.2722224 0.01703538 NA
## 23 0.1144166 0.0008406108 0.01896374   0.2722224 0.01703538 NA
## 24 0.1204064 0.0008672240 0.01728061   0.3067304 0.01781600 NA
## 25 0.1204064 0.0008672240 0.01728061   0.3067304 0.01781600 NA
```

If you are interested in a specific parameter, for example 'psi', you can load the posterior samples for that specific parameter as a vector with `loadparambyname`.

```
psiposterior <- loadparambyname(outputfolder, burn_in, 'psi')
plot(density(psiposterior), main = 'Psi posterior sample density')
polygon(density(psiposterior), col = 'orange')
```



Conclusion

The above covers the basics of parsing EpiFusion output, and introduces some of the functions of EpiFusion_utilities.R. We stress that EpiFusion is still in its infancy, with more improvements happening every day, but we hope this tutorial will have peaked your interest! For more information we recommend checking out the EpiFusion wiki.