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

Appendix 1 - Creating A Symbolic Link

By creating a symbolic link, it is possible to use the EpiFusion command line software from anywhere on your system. Below we provide structions on this process:

1. Download an EpiFusion jar file and place it somewhere safe on your system, for example `/usr/local/bin`.
2. Open a terminal window and type
```
echo $PATH
```
3. Navigate to one of the filepaths that appears.
4. Create a bash script that will run EpiFusion and pass arguments to it. You can do this with a text editor like 'nano' or 'vim'. The bash script can be called anything, but it should contain the following content:
```
#!/bin/bash
java -jar /absolute/path/to/EpiFusion/jar/file.jar "$@"
```
5. Create a symbolic link between this script and the keyword 'EpiFusion'. You can do that by typing the following command:
```
ln -s /path/to/wherever/the/bash/script/was/made EpiFusion
```

Now you should be able to call EpiFusion from anywhere in your system.

Appendix 2 - Full EpiFusion XML Breakdown

Below we provide information on the input files for EpiFusion (XML files). For some key parameters and priors, we provide advice on setting reasonable parameter values or priors.

Loggers

The loggers section dictates the program output,

`fileBase`

The name of the folder to be created for for output files (`<fileBase>`). Unless you specify a filepath in `fileBase`, the output files will be written to your working directory. This can be anything, but avoid the usual folder naming trouble makers (spaces, slashes, etc).

`logEvery`

This sets the frequency at which the program logs the state of the MCMC to the output files (and prints to console). If you are concerned about autocorrelation (MCMC samples close by in the chain having very similar values, indicating a poorly explored posterior), we recommend increasing this parameter until you are satisfied with the mixing of your chains. You may have to run your chains for a longer number of steps in this instance to ensure you still get a satisfactory Effective Sample Size of samples from the posterior.

Data

The data section is for providing case incidence data, a phylogenetic tree, and how you wish to weight their contribution to the model.

`Incidence`

You can provide incidence to the model inside the `<incidence>` tag.

```
<incidence>
 <incidenceVals>1 3 1 5 18 35 40 25 24 13 2 0 1 0</incidenceVals>
 <incidenceTimes type = "every">7<incidenceTimes>
</incidence>
```

`IncidenceVals`

The values go in the `<incidenceVals>` tag, ordered forwards in time from the beginning of the modelled time period

`IncidenceTimes`

The `incidenceTimes` tag can be used to specify the times of incidence observations forwards in time from the beginning of the model time period. There are a few options for specifying

the times with the `type` attribute inside the tag - either specifying the interval between observations or specifying exact times that incidence was recorded:

```
<incidenceTimes type = "every">7<incidenceTimes>
```

For example above we specify that incidence is recorded every 7 days, starting on day 7.

```
<incidenceTimes type = "exact">10 14 25 31 56<incidenceTimes>
```

And here we provide the exact times that incidence data was recorded. Note that if you use this option, the length of `incidenceTimes` should match `incidenceValues`.

`Tree`

Provide a timescaled phylogenetic tree (or tree posterior, for example from a BEAST analysis) to the model using the `<tree>` tag. Currently, it is necessary to have node a leaf labels formatted ending with square brackets with the node times in these brackets (e.g. `node_1[0.3442]`). This requires some preprocessing, we recommend using the `prepare_epifusion_tree()` function in `EpiFusionUtilities` to do this conversion for you.

```
  <tree>
    <treePosterior>false</treePosterior>
    <treeFile>Data/Processed/processed_fixed_tree.tree</treeFile>
  </tree>
```

You may either include the tree string in the XML file using the `treeString` tag, or point to a file containing the tree(s) using `treeFile`

`treeString`

This option is only available if you are using a single fixed tree, where inside the `tree` tag you can include the tree string inside `treeString` tags.

```
<treeString>(((((((leaf_407[70.10236230370634]:19.09030831,leaf_377[71.67903
395633373]:20.66697996)node_23[51.0120539946585]:16.72849832,(leaf_530[64.290
19242068087]:28.60128838,(((leaf_678[55.836692533807465]:12.44871907,leaf_692
[54.95812431493469]:11.57015085)node_52[43.38797346813441]:0.385599307,leaf_6
40[58.57019798061809]:15.56782382)node_51[43.00237416117269]:4.149624169,leaf
_223[80.20988528843057]:41.3571353)node_49[38.852749991991104]:3.163845956)no
de_32[35.68890403623855]:1.405348363)node_9[34.28355567352641]:5.723132688,((
leaf_242[78.62221627030786]:37.6032997,((leaf_153[85.38189406711462]:30.46531
99,leaf_75[95.31697617294901]:40.400402)node_112[54.91657417003654]:9.0366305
64,leaf_111[90.02540774077697]:44.14546413)node_108[45.879943605975534]:4.861
027036)node_87[41.018916570389365]:6.268352098,((leaf_28[110.31768492437661]:
56.8016204,leaf_445[68.36948591902012]:14.8534214)node_161[53.51606452349725]
:6.970120811,leaf_752[48.45392169632879]:1.907977984)node_160[46.545943712076
95]:11.79537924)node_84[34.750564472466806]:6.190141487)node_8[28.56042298562
4168]:4.876221701,(((((leaf_74[95.8576156329268]:37.00079705,leaf_396[70.5595
5806481323]:11.70273948)node_192[58.85681858553985]:10.32855116,leaf_662[57.0
72280245746875]:8.544012822)node_190[48.52826742337909]:5.282985936,leaf_109[
```

90.08763067992984]:46.84234919)node_187[43.24528148741703]:6.403050181,leaf_102[90.85197148216028]:54.00974018)node_185[36.84223130663702]:1.46060558,((leaf_70[96.23083229433816]:57.38098621,leaf_596[60.77116595386879]:21.92131987)node_223[38.84984607971327]:0.1026768434,(leaf_159[84.84128196112528]:30.74399116,leaf_166[84.10666840469251]:30.0093776)node_240[54.0972908003288]:15.35012156)node_222[38.74716923627249]:3.36554351)node_176[35.38162572640305]:11.69742444)node_7[23.68420128510857]:11.46063414,(((leaf_160[84.82844619112666]:60.75968714,((((leaf_64[98.73242319027953]:58.21388631,leaf_681[55.580816911770 14]:15.06228003)node_294[40.51853687799632]:1.735152486,leaf_590[61.148684429556546]:22.36530004)node_293[38.783384391686475]:0.551139453,leaf_98[91.67047000399187]:53.43822507)node_292[38.23224493867514]:10.97801371,leaf_791[43.377517040053526]:16.12328581)node_289[27.25423123067509]:3.185472183)node_248[24.068759047729554]:0.4997283961,(leaf_553[63.34492352254521]:28.29037819,leaf_99[91.60953189160091]:56.55498656)node_329[35.05454533491508]:11.48551468)node_247[23.56903065162453]:7.719268377,((leaf_309[74.58310015128357]:13.365411254,leaf_55[101.81985847090435]:40.60217086)node_362[61.21768760926006]:34.52413937,leaf_424[69.34790109229266]:42.65435285)node_359[26.69354824266421]:10.84378597)node_246[15.849762274390272]:3.62619513)node_5[12.223567144247692]:3.195696408,(((leaf_162[84.40937076572243]:31.50911351,leaf_324[73.97434087205923]:21.07408361)node_395[52.90025725951645]:9.663695667,leaf_507[65.4925795095511]:22.25601792)node_388[43.23656159282253]:12.85871416,(((leaf_667[56.73582187536703]:2.027004712,leaf_271[76.9847635381094]:22.27594637)node_442[54.70881716344273]:1.414390225,leaf_366[72.0731573444004]:18.77873041)node_441[53.294426938013345]:12.38434243,leaf_292[75.68955369039529]:34.77946918)node_431[40.91008450554362]:10.53223707)node_385[30.377847436622286]:21.3499767)node_4[9.027870735773057]:2.202856024,(((((((leaf_59[100.08067285788199]:50.16113194,(leaf_122[88.25148926376218]:36.51609694,leaf_481[66.5178885482595]:14.78249623)node_490[51.73539232236806]:1.815851402)node_488[49.91954092033317]:5.658838175,leaf_783[44.504551838421605]:0.2438490932)node_485[44.260702745210416]:3.834595734,leaf_718[52.73400913067007]:12.30790212)node_482[40.42610701140656]:0.3727987892,leaf_137[86.87820106224541]:46.82489284)node_481[40.053308222256206]:1.541827426,leaf_679[55.73806406067391]:17.22658326)node_480[38.51148079662553]:11.31956052,((leaf_415[69.74802935404291]:24.16809076,leaf_542[63.98421692880167]:18.40427833)node_534[45.579938595143936]:16.55024436,leaf_494[65.88016294241126]:36.8504687)node_525[29.02969423779227]:1.837773964)node_469[27.19192027355947]:4.784286462,(((((leaf_240[78.7251493051829]:36.09533547,leaf_733[51.27464829974004]:8.644834469)node_572[42.62981383087519]:3.351534295,leaf_565[62.54393520248931]:23.26565567)node_571[39.27827953617371]:10.94774507,((((leaf_544[63.77673952279264]:26.31061793,leaf_363[72.15967419706526]:34.69355261)node_611[37.46612159103572]:0.04411065085,leaf_41[105.11850117358156]:67.69649023)node_610[37.4220109401828]:2.236590466,((leaf_505[65.51898007568552]:14.55632746,leaf_452[67.98768873895207]:17.02503612)node_662[50.96265262067672]:1.536519265,leaf_466[67.2387130185725]:17.81257966)node_661[49.42613335594053]:14.24071288)node_609[35.18542047400971]:4.220130407,leaf_774[45.24321718225812]:14.27792712)node_606[30.965290067180977]:2.634755598)node_570[28.330534468731727]:0.9269283664,(((((leaf_140[86.67125153426835]:46.3559461,leaf_524[64.58552017616272]:24.27021474)node_700[40.315305433596436]:2.215938869,((leaf_334[73.56013467469849]:10.73098258,leaf_517[65.14498837179052]:2.315836279)node_731[62.829152092327995]:15.4593817,leaf_307[74.69334748527966]:27.32357709)node_728[47.36977039079453]:9.270403827)node_698[38.09

```
9366564151296]:3.31115576,leaf_444[68.42688854802711]:33.63867774)node_697[34
.78821080397044]:2.208122614,(leaf_564[62.62160623104026]:17.51198355,leaf_76
3[47.149467892186756]:2.039845215)node_744[45.109622677313816]:12.52953449)no
de_696[32.580088189594434]:3.733445308,(((leaf_63[99.03748147442674]:36.16276
457,leaf_183[82.93567799222579]:20.06096108)node_760[62.87471690918196]:16.65
804619,(leaf_65[97.92430834876784]:41.86612862,leaf_209[80.97839091724356]:24
.92021119)node_762[56.058179730263774]:9.84150901)node_758[46.216670720407905
]:13.3159056,leaf_811[36.537443151630136]:3.63667803)node_754[32.900765121763
64]:4.05412224)node_694[28.846642881868824]:1.44303678)node_569[27.4036061023
49244]:1.228209889,((leaf_319[74.20982901626131]:24.16887906,(leaf_492[66.036
2745797389]:12.06101312,(leaf_451[68.04527961439327]:10.70473666,leaf_487[66.
15872756629061]:8.818184615)node_793[57.34054295105745]:3.365281488)node_790[
53.975261462789035]:3.934311504)node_776[50.04094995889265]:4.478634208,leaf_
94[92.54528570216107]:46.98296995)node_775[45.562315751008036]:19.38691954)no
de_568[26.175396213176917]:3.767762401)node_466[22.40763381199003]:2.28485233
2,(leaf_670[56.498190314744825]:6.555477923,leaf_304[74.78607366094431]:24.84
336127)node_811[49.94271239223166]:29.81993091)node_465[20.122781479584827]:1
3.29776677)node_3[6.825014711946831]:0.3910339013,leaf_212[80.7513012802764]:
74.31732047)node_2[6.433980810655798]:6.433980810655798;
    </treeString>
```

treeFile

An alternative option to `treeString` is `treeFile` where you provide the path from your working directory to a file with the tree or tree posterior.

```
<treeString>Data/Processed/processed_fixed_tree.tree</treeString>
```

treePosterior

If you are passing a tree posterior to EpiFusion, this should be set to true. If you are generating your XML file using the `EpiFusionUtilities` function `generate_epifusion_xml`, however, the function will automatically detect whether a tree or tree posterior has been provided and will set this for you.

Weighting

The data block also allows you to set the 'influence' each set of data has over the inferred epidemic trajectories, using the `epicontrib` and `changetimes` parameters, however ***we currently do not recommend setting these parameters as anything but the defaults (0.5 and 0)***. These default values essentially make it so that each dataset is contributing equally, and this is all we have validated the model for so far.

```
<epicontrib>0.5</epicontrib>
<changetimes>0</changetimes>
```

Analysis

The analysis block is another block you probably won't have to change from the default. It will usually look like this:

```
<analysis>
  <type>looseformbeta</type>
  <startTime>null</startTime>
  <endTime>null</endTime>
  <inferTimeOfIntroduction>false</inferTimeOfIntroduction>
</analysis>
```

Most analyses will use `looseformbeta` as the type. This means you don't want to set any expectations for the movements beta (the daily rate of infection) will make. So, each particle in the particle filter will each have their own beta, which will vary over time in a random walk and yield a unique beta trajectory (you can parameterise the nature of this random walk in `priors`).

You can also specify the `start` and `end` time of the analysis (in days) if you have data covering a timeframe longer than what you are interested in - this is mainly for epi only analyses.

Finally, if you wish to set a flexible start date in your model, i.e. infer the date of outbreak origin, `inferTimeOfIntroduction` should be set to true. Note that if this is the case, you will need to set a prior for `outbreakOrigin` in `priors`.

Model

In this block you can customise the model a bit more - as we add improvements to EpiFusion we intend to make it flexible and modular to allow more user control over the model. For now, you can use this block to decide between epidemiological observation models. There are currently two options:

Poisson

```
<model>
  <epiObservationModel>poisson</epiObservationModel>
</model>
```

Negative Binomial

The negative binomial can be used as a special case of the poisson to account for overdispersion in your data. Currently the overdispersion parameter must be set manually - eventually we hope to offer the ability to infer it.

```
<model>
  <epiObservationModel>negbinom</epiObservationModel>
  <overdispersion>10.0</overdispersion>
</model>
```

Parameters

There are **many** of parameters to choose from for your analysis, and this and the priors block are the ones you will likely be editing the most - so get familiar. Here's what it looks like:

```
  <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>
```

Now let's go through each component in turn.

`epiOnly`

Set it to true if you want to run an case incidence only analysis. If it, and `phyloOnly` are both `false`, a combined analysis will be run.

`phyloOnly`

Set it to true if you want to run a tree only analysis. If it, and `epiOnly` are both `false`, a combined analysis will be run.

`numParticles`

Does what it says on the tin: numParticles sets the number of particles for the particle filter. 100-200 particles usually works fine, if you are expecting unpredictable movements it's usually helpful to aim towards the higher end. For very long time series' or analyses with many 'twists' and 'turns' in infection dynamics, it is recommended to increase this number, but this will lead to a decrease in efficiency. If your acceptance rate is very low, increasing the number of particles may also help.

`numSteps`

Number of steps of the MCMC. The particle filter does a lot of heavy lifting fitting wise, so 2000-5000 steps often does the trick. However for complex analyses, or analyses with low acceptance rates, it may be advisable to run for longer. It is important to check your results to ensure you have enough steps in each chain for them to converge on the posterior (you

can verify this visually using trace plots or by checking the gelman-rubin statistics of the parameters), and yield satisfactorily high Effective Sample Sizes of the MCMC parameters.

`numThreads`

Number of threads to use during the parallelised particle filter steps. More threads will lead to a faster runtime - but try not to assign more threads than are available on your machine.

`numChains`

Number of MCMC chains to run. Standard practice is around 4. Sometimes chains get stuck and their MCMC step acceptance rate drops to zero. If you are worried about this, you can increase the number of chains you run to allow you to discard non-convergent or stuck chains. If your chains are commonly getting stuck or not converging, you should closely examine your data and analysis to ensure there are not any bigger problems at play.

`stepCoefficient`

Coefficient of the MCMC step sizes for generating new parameter proposals. Currently the MCMC sampler is basic Metropolis Hastings so you might have to mess around with this step-size to find what works best. Anecdotally, we have found between 0.01 and 0.1 to be successful at converging on the posterior while not being too large so that every step is rejected.

`resampleEvery`

This sets the number of days between each particle resampling. 7 days has worked nicely for us so far, and gels nicely with weekly case incidence counts. Increasing this value (i.e. making the intervals larger) helps with efficiency, but increases the risk of particle depletion and decreases the resolution of the inferred dynamics if you are using a linear splines approach to fitting beta.

`segmentedDays`

This makes a minor change to the phylogenetic likelihood implementation. When it is `false`, the phylogenetic weighting is done in discrete daily increments, but if you have a tree with a lot happening (multiple internal or external nodes per day) then `<segmentedDays>true</segmentedDays>` breaks down the days into slightly finer chunks which can help with your MCMC acceptance rate when prevalence is low. It's generally better safe than sorry to keep segmentedDays as true, it just marginally slows down the analysis (very slightly).

`samplingsAsRemovals`

If you are modelling sampling events on the tree as removals from the infected population, this should be 1. Otherwise set it to 0.

`pairedPsi`

For analyses where the sampling rate needs to be carefully parameterised and perhaps vary over time, sometimes it is more logical to infer only the case sampling rate and calculate the proportional genomic case sampling rate (`psi`) from the data using the ratio of observed sequences in the tree over time to observed epidemiological cases. Setting this parameter to true introduces this option.

Priors

The priors block is where you provide priors and other parameterisations for the rates of the particle filter process model. You can provide a number of different priors, but you definitely must include a `gamma`, `phi` and `psi` (unless you're using the `pairedPsi` feature). All priors need two things: whether or not the rate has a step-change (`stepchange`), and then details on how the rate is distributed (`disttype` and associated info). For example, here is a gamma parameterisation where the rate stays constant over time (no step-change) and a truncated normal distribution with mean 0.143, standard deviation 0.03 and lower bound of 0.0.

```
    <gamma>
        <disttype>TruncatedNormal</disttype>
        <mean>0.143</mean>
        <standarddev>0.03</standarddev>
        <lowerbound>0.0</lowerbound>
        <stepchange>false</stepchange>
    </gamma>
```

Distribution Options

There are a number of different distribution options available with more being added all the time. Here is what's currently available: 1. Normal 2. Truncated Normal 3. Poisson 4. Uniform 5. Uniform Discrete 6. Beta 7. FixedParameter (this is if you want to fix a parameter, i.e. not infer it)

And here's an example of how each of them could be parameterised:

```
    <exampleparam>
        <disttype>Normal</disttype>
        <mean>0.143</mean>
        <standarddev>0.03</standarddev>
        <stepchange>false</stepchange>
    </exampleparam>
```

```
    <exampleparam>
        <disttype>TruncatedNormal</disttype>
        <mean>0.143</mean>
        <standarddev>0.03</standarddev>
        <lowerbound>0.0</lowerbound>
        <stepchange>false</stepchange>
    </exampleparam>

    <exampleparam>
        <disttype>Poisson</disttype>
        <mean>500</mean>
        <stepchange>false</stepchange>
    </exampleparam>

    <exampleparam>
        <disttype>Uniform</disttype>
        <min>0.1</min>
        <max>0.8</max>
        <stepchange>false</stepchange>
    </exampleparam>

    <exampleparam>
        <disttype>UniformDiscrete</disttype>
        <min>400</min>
        <max>600</max>
        <stepchange>false</stepchange>
    </exampleparam>

    <exampleparam>
        <disttype>Beta</disttype>
        <alpha>7</alpha>
        <beta>1000</beta>
        <stepchange>false</stepchange>
    </exampleparam>

    <exampleparam>
        <disttype>FixedParameter</disttype>
        <value>0.15</value>
        <stepchange>false</stepchange>
    </exampleparam>
```

Setting reasonable priors

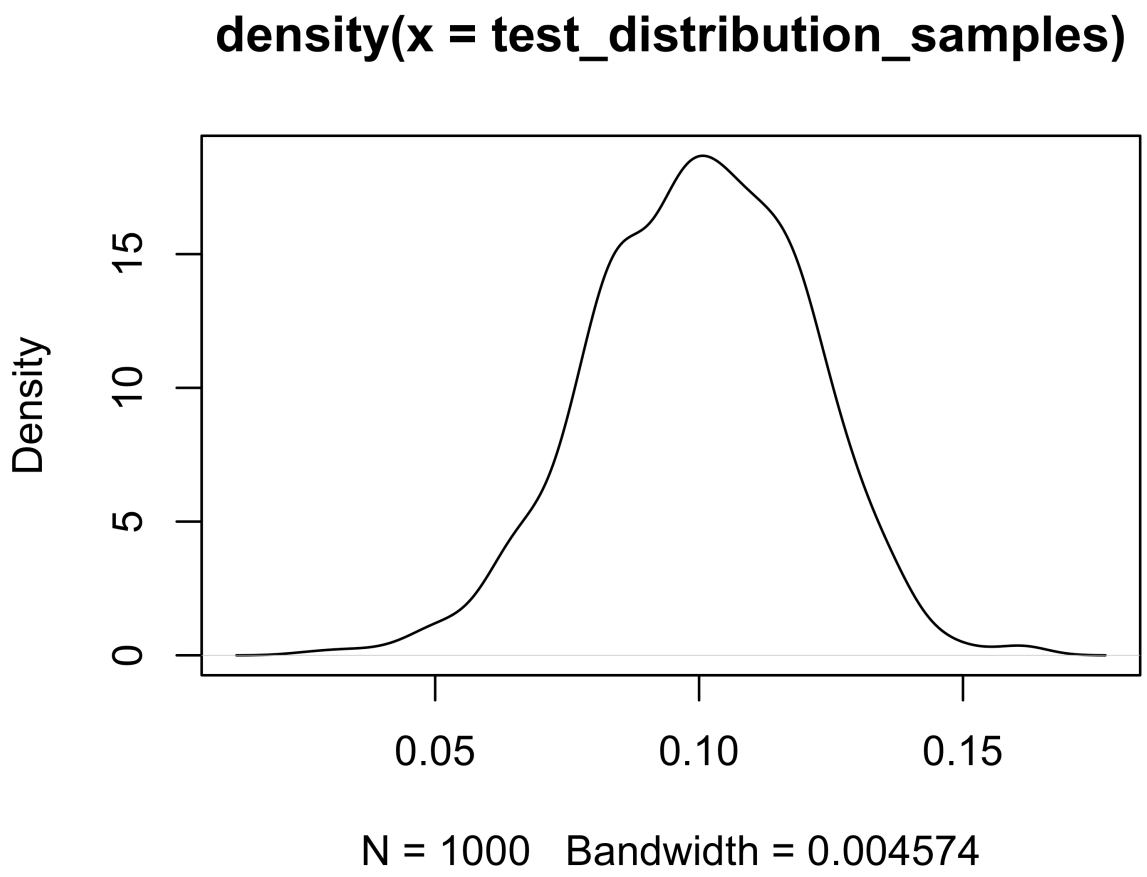
When setting priors for EpiFusion we recommend using some trial and error until you find the values which make sense for your analysis. It is also sometimes useful to visually inspect the distributions you are specifying by plotting the values obtained from functions such as `rnorm`, `rtruncnorm`, `rbeta` etc. in an R session, and ensure that the 'parameter' values along

the x-axis with the highest values along the y-axis correspond to your prior beliefs about the parameter value.

For example, here we inspect the shape of a proposed prior for `gamma` for a disease with an approximately 10 day infectious period:

```
test_distribution_samples <- rtruncnorm(1000, a = 0, mean = 0.1, sd = 0.02)

plot(density(test_distribution_samples))
```



Most priors set in EpiFusion are for rates, meaning they should be exclusively positive and parameterised with distributions such as TruncatedNormal (with a lowerbound of 0.0) or Beta. For distributions that require a mean, this value should be close to your suspected value of the parameter according to your existing knowledge. The width of the distribution should reflect your certainty about the value of the parameter - this is often governed by the standard deviation parameter, or the minimum and maximum parameters for uniform distributions. If you have very little knowledge or confidence in the value of the parameter, uniform distributions may work best, as they are 'uninformative'; i.e. they do not place any

restrictions on the values of the parameter, except perhaps their minimum and maximum values.

Gamma

This is the rate of recovery or removal of an infectious individual. For example, if the disease under study has an infectious period of 10 days, then a reasonable mean value for your prior distribution will be 1/10, or 0.1.

Beta

While itself is typically fit within the particle filter, many of these fitting methods allow you to infer the starting point via MCMC. The parameter is the infectivity of an infectious individual, and is used with to infer R_t from the model using the equation $R_t = \beta/\gamma$. For example, if you expect R_t to be approximately 3.0 at the beginning of the time series, and you have parameterised with a mean of 0.1 as per our example above, then you might parameterise with a mean of $3.0 * 0.1 = 0.3$.

Phi and psi

and represent the case and genomic sequence sampling rates, respectively - i.e. what the daily likelihood is of an infectious individual being sampled as a case or sequence in the data. If you have provided and tree and case incidence data to the model, these must be parameterised in some way. They are naturally somewhat paired, i.e. the number of genomes is typically some smaller proportion of the number of cases. Accordingly their prior parameterisation should reflect this relationship. Alternatively, it is possible to pair them within the mode, and only specify a prior for the parameter.

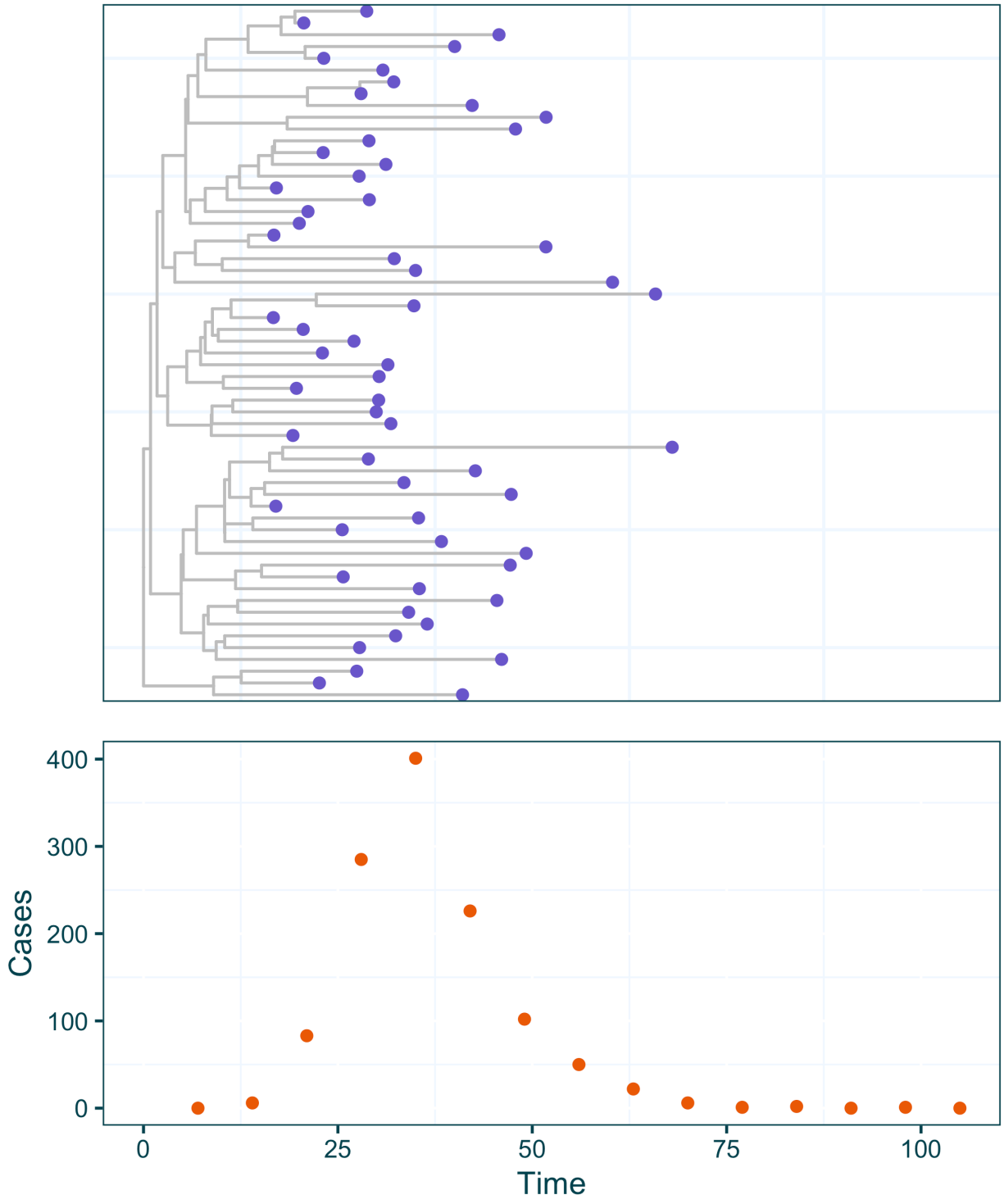
Appendix 3 - Paired Sampling Infrastructure

Specifying the case and genomic sequence sampling rates for EpiFusion is an important aspect of the parameterisation, however it should be noted that the these should be somewhat linked i.e. their proportion of the total infections will be related to their proportion of each other. With this in mind it is possible to parameterise the `psi` (genomic sequencing proportion of all infections) parameter as being linked to the `phi` parameterisation, where the fraction of sampled sequences as a proportion of cases over time is used in conjunction with the `phi` parameterisation, in place of specifying these parameters separately.

Consider an analysis with the following data chunk in it's XML file:

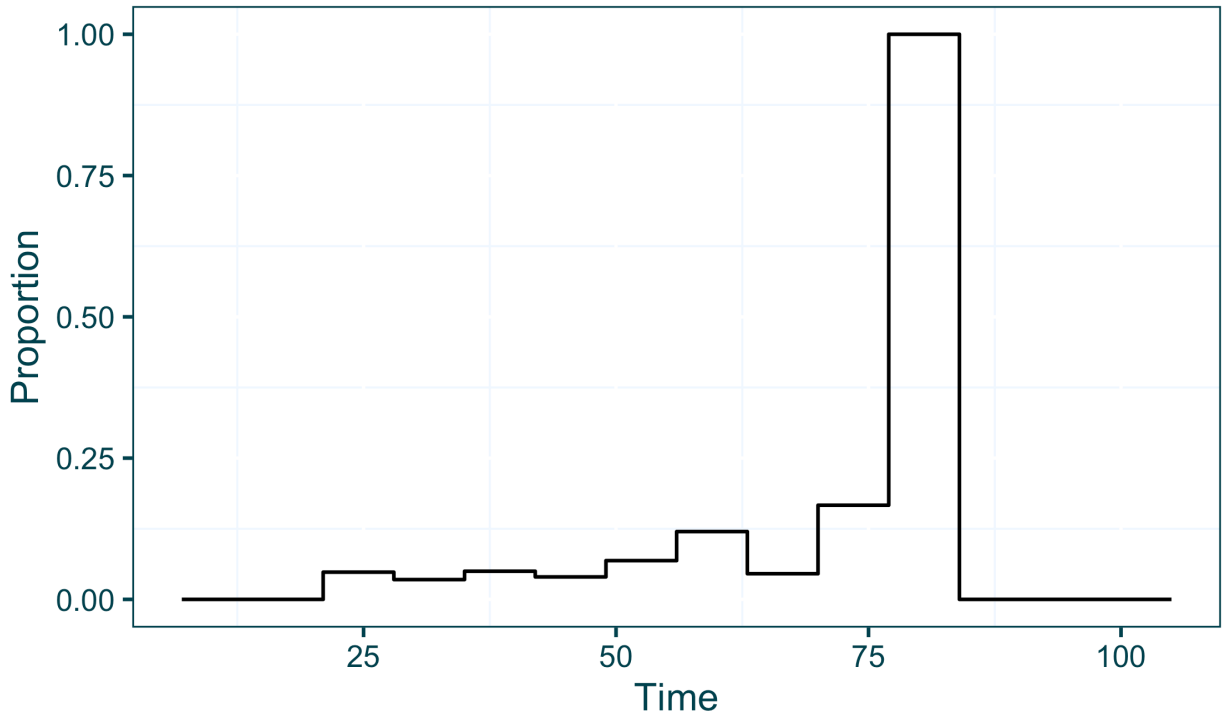
```
<data>
  <incidence>
    <incidenceVals>0 6 83 285 401 226 102 50 22 6 1 2 0 1 0</incidenceVals>
    <incidenceTimes type="every">7</incidenceTimes>
  </incidence>
  <tree>(((leaf_978[24.289092735613732]:10.05812796,leaf_751[29.0936183621188
2]:14.86265359)node_14[14.230964773732369]:3.553266042,leaf_172[42.6977273960
0777]:32.02002866)node_4[10.67769873203644]:9.009220774,(((((leaf_727[29.4567
32285001838]:17.34678626,leaf_444[34.098590228053204]:21.9886442)node_160[12.
109946023683246]:1.093924779,leaf_98[47.72409501869759]:36.70807377)node_159[
11.016021244423117]:1.619071921,(leaf_292[38.15303138198247]:28.16886388,(lea
f_379[35.76908018744906]:21.99507627.........node_968[7.674601453741431]:0.59
00236866)node_900[7.084577767125148]:2.936346776,(leaf_11[61.99365320703872]:
56.29610808,((leaf_348[36.65407632226596]:24.86354942,leaf_457[33.91669350511
818]:22.1261666)node_1100[11.790526905301487]:3.459163323,(leaf_46[53.4250715
4674088]:38.25619549,leaf_1146[18.439048081586712]:3.270172027)node_1152[15.1
68876054251774]:6.837512472)node_1099[8.331363582073376]:2.633818457)node_103
9[5.6975451254480305]:1.549314134)node_890[4.14823099105802]:0.7375491204)nod
e_511[3.4106818706794892]:0.8417615721)node_97[2.568920298626249]:0.900442340
4)node_1[1.6684779582722558]:1.6684779582722558;</tree>
  <epicontrib>0.5</epicontrib>
  <changetimes>0</changetimes>
</data>
```

This data chunk inputs the below data into EpiFusion (*Supplementary Figure S1*) - weekly case incidence (totalling 1185 cases), and a phylogenetic tree of sequenced samples (59 sequences in total).



Supplementary Figure S1 An example dataset suitable for use with EpiFusion, with a phylogenetic tree made up of genomic sequences sampled from an outbreak, and weekly case incidence.

If we look at the number of sampled sequences (tree tips) over time as a proportion of the corresponding cases, we get something that looks like this:



Supplementary Figure S2 Proportion of genomic sequences to case incidence data points from the dataset, calculated in intervals between each case incidence data point.

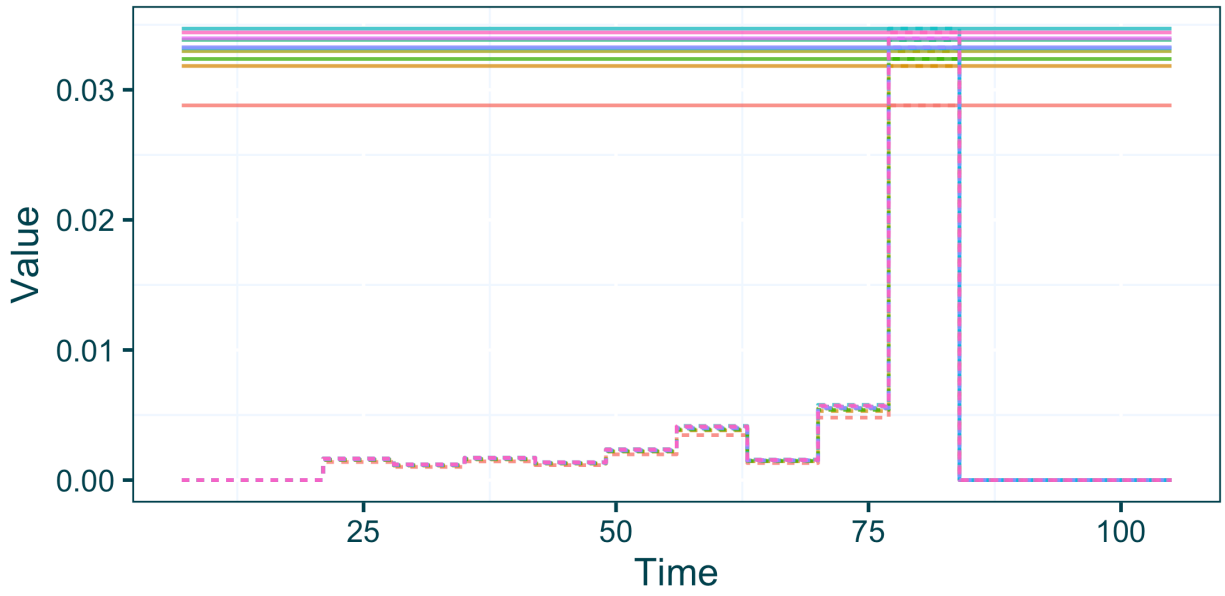
Pairing the psi and phi parameters

In a situation like this where the user doesn't want to provide priors for the `psi` parameter specifically, the XML file can be adjusted to contain the following in place of a `<psi>` block within the `<priors>` section. Below, the `pairedPsi` tag indicates that the user is not specifically parameterising `psi`, but instead it will be calculated using the parameterisation of the `phi` case sampling rate and the proportion of sequences to cases as which is calculable from the data.

```
<priors>
  <gamma>
    <!-- gamma parameterisation goes here -->
  </gamma>
  <pairedPsi></pairedPsi>
  <phi>
    <!-- phi parameterisation goes here -->
  </phi>
  <initialBeta>
    <!-- initalBeta parameterisation goes here -->
  </initialBeta>
  <betaJitter>
    <!-- betaJitter parameterisation goes here -->
  </betaJitter>
</priors>
```

Effect in the MCMC sampling process

The resulting effect in the MCMC sampling process within EpiFusion is that the `phi` case sampling rate is fitted as a normal MCMC parameter, and the `psi` sequence sampling rate is fitted accordingly as a proportion of `phi` across time calculated from the data. This is demonstrated below, where 10 samples of the `phi` sampling rate and their corresponding `psi` values are plotted. Different colours indicate different MCMC samples, with the solid line showing `phi` and the dashed line showing the corresponding `psi`.



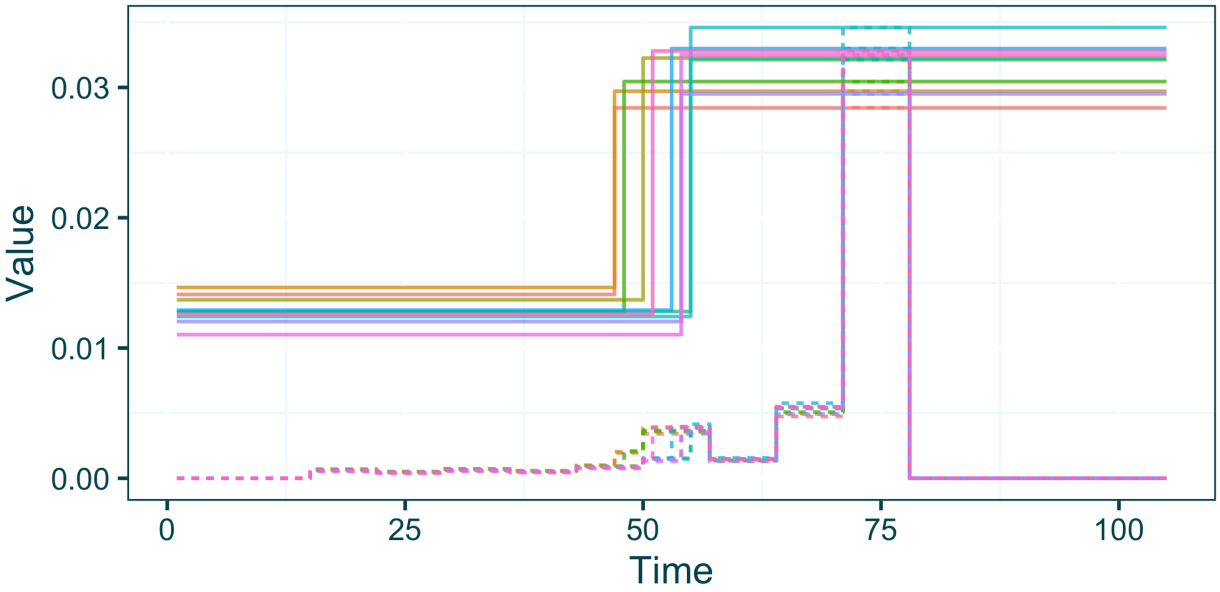
Supplementary Figure S3 Plot of case incidence (solid line) and genomic sequence (dashed line) sampling rates, sampled from the EpiFusion MCMC. The straight horizontal coloured lines represent a sampled value of the case sampling rate, and the dashed line in the corresponding colour represent the paired genomic sampling rate calculated using the proportion of sequences to cases in the data.

As is evident in the above graph, `phi` is parameterised as a constant rate over time (the solid horizontal lines). To demonstrate the paired psi feature wtih a more advanced parameterisation of `phi`, the following `priors` XML chunk can be used.

```
<priors>
  <gamma>
    <stepchange>false</stepchange>
    <disttype>TruncatedNormal</disttype>
    <mean>0.143</mean>
    <standarddev>0.05</standarddev>
    <lowerbound>0.0</lowerbound>
    <upperbound>1.0</upperbound>
  </gamma>
  <pairedPsi></pairedPsi>
  <phi>
    <stepchange>true</stepchange>
    <changetime>
```

```
      <x0>
        <disttype>UniformDiscrete</disttype>
        <min>45</min>
        <max>55</max>
      </x0>
    </changetime>
    <distribs>
      <x0>
        <disttype>TruncatedNormal</disttype>
        <mean>0.01</mean>
        <standarddev>0.002</standarddev>
        <lowerbound>0.0</lowerbound>
        <upperbound>1.0</upperbound>
      </x0>
      <x1>
        <disttype>TruncatedNormal</disttype>
        <mean>0.03</mean>
        <standarddev>0.01</standarddev>
        <lowerbound>0.0</lowerbound>
        <upperbound>1.0</upperbound>
      </x1>
    </distribs>
  </phi>
  <initialBeta>
    <stepchange>false</stepchange>
    <disttype>Uniform</disttype>
    <min>0.2</min>
    <max>0.9</max>
  </initialBeta>
  <betaJitter>
    <stepchange>false</stepchange>
    <disttype>Uniform</disttype>
    <min>0.001</min>
    <max>0.1</max>
  </betaJitter>
</priors>
```

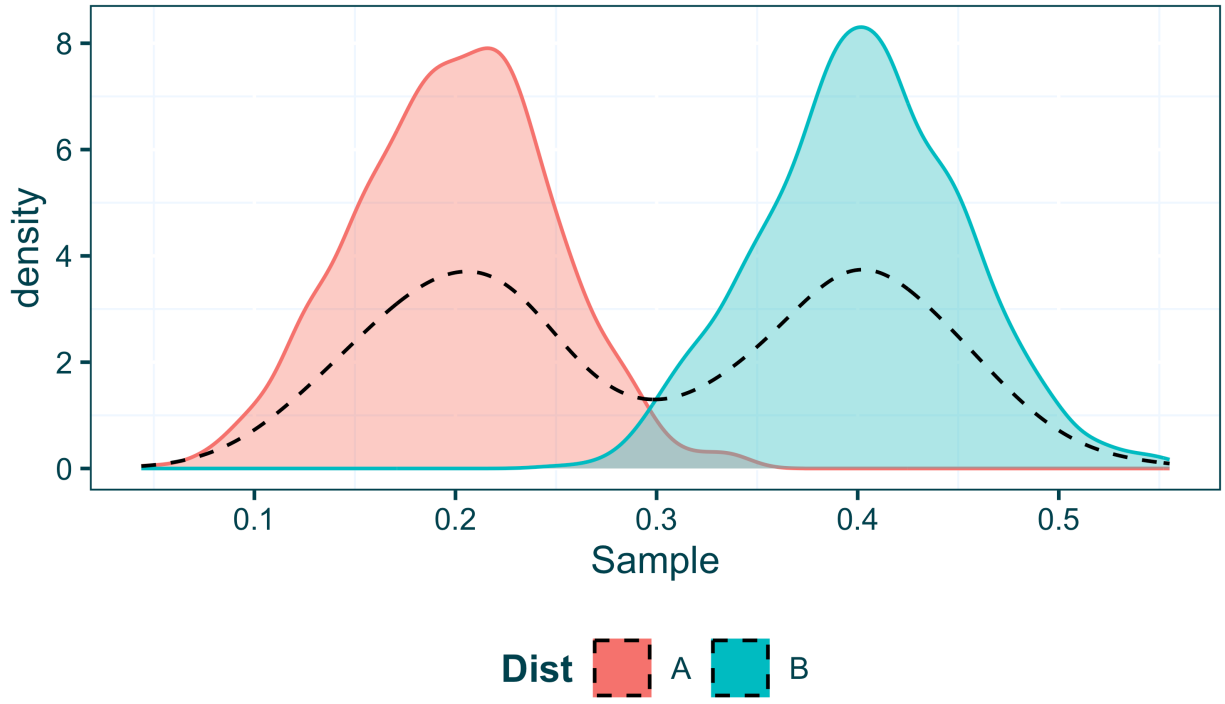
The above amended XML chunk, specifically the `phi` block, parameterises a step-change in the `phi` case sampling rate at some point during the outbreak, with a uniform prior on the exact time of this change being between day 45 and 55. Some resulting `phi` samples, and paired `psi`, look like this.



Supplementary Figure S4 Plot of case incidence (solid line) and genomic sequence (dashed line) sampling rates, sampled from the EpiFusion MCMC. The straight horizontal coloured lines represent a sampled value of the case sampling rate, and the dashed line in the corresponding colour represent the paired genomic sampling rate calculated using the proportion of sequences to cases in the data. In this example, the case sampling rate is also parameterised with a step change, that is reflected in the values over time in both the case sampling rate and the paired genomic sampling rate.

Appendix 4 - Composite Prior Distributions

It is not uncommon to want to specify priors or distributions of variables that have a non-parametric form. To accomodate this we include the option to EpiFusion to specify priors with non-parametric distributions that are the composite of two or more parametric distributions.



Supplementary Figure S5 Example of a bimodal non-parametric probability distribution (black dotted line) which is a composite of two parametric distributions with different means (blue, pink).

Parameterisation in EpiFusion XML

In EpiFusion XML, prior distributions are specified inside the `priors` block, inside XML tags that describe the different variables (most essentially, `gamma`, `psi`, `phi` and some parameterisation of `beta` depending on the fitting method). For a given parameter, e.g. `phi` this can look a little like this:

```
<phi>
      <stepchange>false</stepchange>
      <disttype>TruncatedNormal</disttype>
      <mean>0.2</mean>
      <standarddev>0.05</standarddev>
      <lowerbound>0.0</lowerbound>
</phi>
```

In the above example, our parameterisation of `phi` is constant throughout the time series we are modelling (indicated by `<stepchange>false</stepchange>`). Our prior for phi is a simple truncated normal distribution.

To adjust this so that our prior is made of two composite distributions (like what is shown above) requires two small changes to the xml.

1. Add a new tag to the XML chunk `numdists` specifying the number of distributions involved
2. Enclose the details of the distributions into new 'sublevel' tags labelled with letters of the alphabet

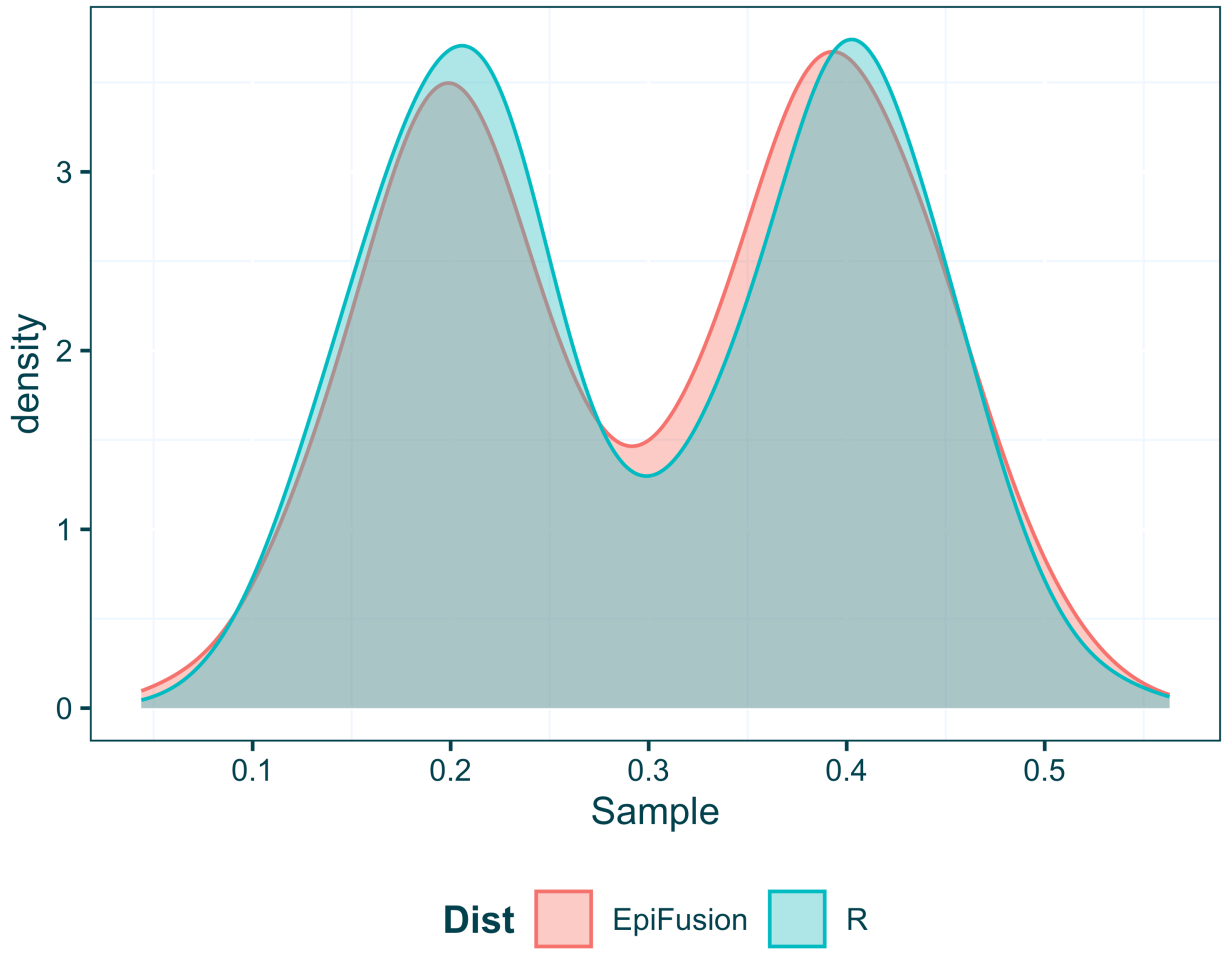
In practice, here's what the adjusted XML should look like:

```
<phi>
  <stepchange>false</stepchange>
  <numdists>2</numdists>
  <a>
    <disttype>TruncatedNormal</disttype>
    <mean>0.2</mean>
    <standarddev>0.05</standarddev>
    <lowerbound>0.0</lowerbound>
  </a>
  <b>
    <disttype>TruncatedNormal</disttype>
    <mean>0.4</mean>
    <standarddev>0.05</standarddev>
    <lowerbound>0.0</lowerbound>
  </b>
</phi>
```

Above we specify that the `phi` prior is made up of two distributions, (`<numdists>2</numdists>`), which we have enclosed in chunks `<a>` and `<b>`.

Validation by sampling from EpiFusion distribution

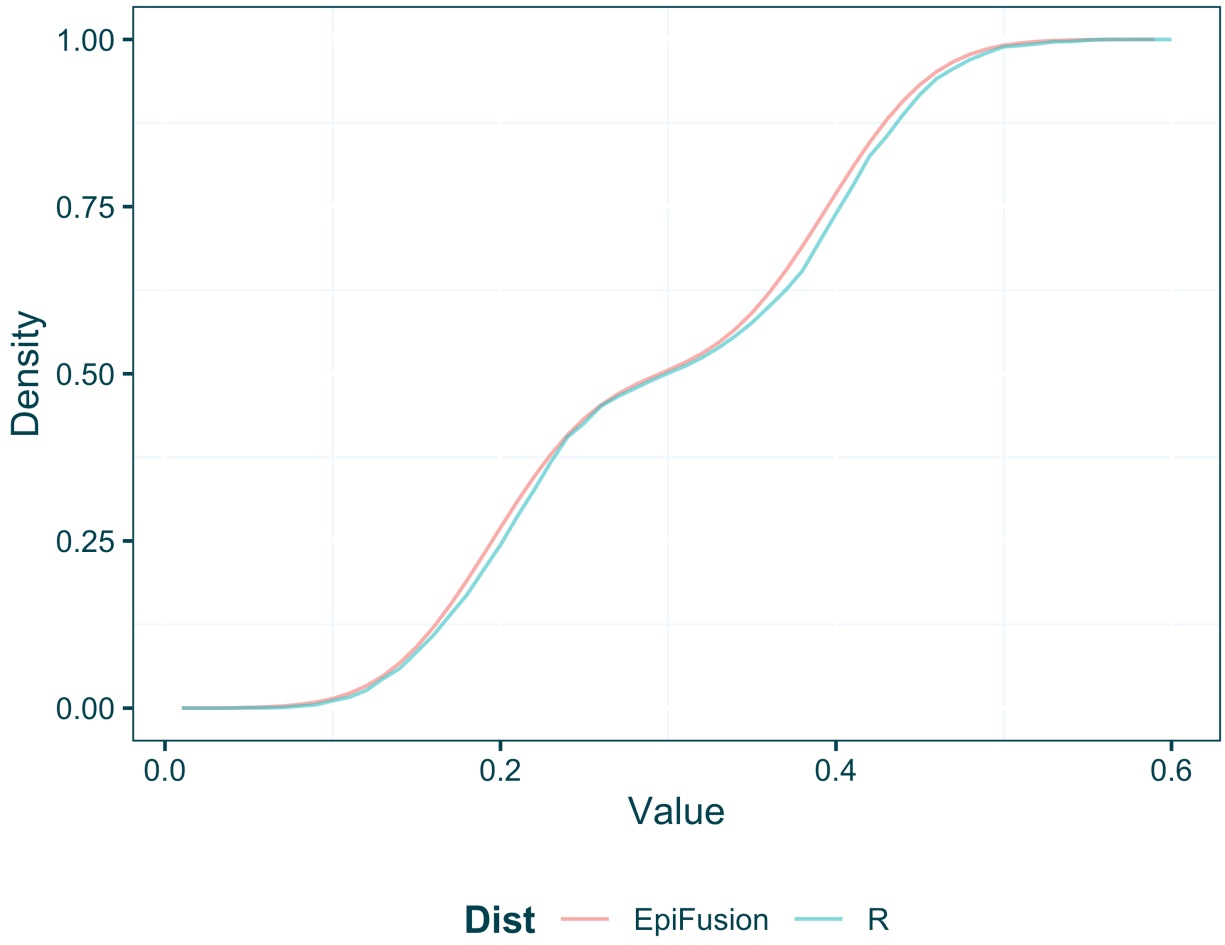
We generated 1000 samples from an EpiFusion prior distribution parameterised with the above XML code to verify that the expected distribution shape was obtained. Below we show these samples plotted with the samples generated for the above plot.



Supplementary Figure S6 Samples from an EpiFusion prior distribution composed of two parametric (normal) distributions (pink) compared to the same distribution in R (blue).

Calculating the Prior Probability from the Composite Distribution

To ensure that the prior probability of this composite distribution is accurately calculated within EpiFusion for the Metropolis Hastings Algorithm, we calculate below the ECDF of the distribution generated by R, and compare it to the values obtained from EpiFusion for the encoded composite distribution.



Supplementary Figure S7 ECDF of the composite distributions in Supplementary Figure S6.

Appendix 5 - Options for Fitting Beta

An important aspect of the EpiFusion model is the particle filter which is used to fit β_t, or the force of infection over time. Together with the fitted γ value(s), β_t is used to yield the model R_t trajectory estimates. The default approach for fitting β_t which is included in the template populated by `EpiFusionUtilities` function `generate_epifusion_xml` is to use a random walk within the particle filter.

We provide a number of different options for fitting $beta_t$, using (i) a random walk within the particle filter, (ii) linear splines within the particle filter, (iii) MCMC fitting in epochs by fixing or fitting change times and interval values, or (iv) MCMC fitting the parameters of a logistic function which defines beta over time.

Setting the Beta Approach

The approach for fitting β_t is specified to the program in the `type` node of the `analysis` block using the keywords `looseformbeta`, `linearsplinebeta`, `fixedbeta` or `invlogisticbeta` (for options (i)-(iv) above, respectively).

```
<analysis>
  <type>looseformbeta</type>
  <startTime>null</startTime>
  <endTime>null</endTime>
  <inferTimeOfIntroduction>false</inferTimeOfIntroduction>
</analysis>

<analysis>
  <type>linearsplinebeta</type>
  <startTime>null</startTime>
  <endTime>null</endTime>
  <inferTimeOfIntroduction>false</inferTimeOfIntroduction>
</analysis>

<analysis>
  <type>fixedbeta</type>
  <startTime>null</startTime>
  <endTime>null</endTime>
  <inferTimeOfIntroduction>false</inferTimeOfIntroduction>
</analysis>

<analysis>
  <type>invlogisticbeta</type>
  <startTime>null</startTime>
  <endTime>null</endTime>
  <inferTimeOfIntroduction>false</inferTimeOfIntroduction>
</analysis>
```

Parameterising the Beta Approach

Once the β_t fitting approach is set, its characteristics also must be parameterised in the `priors` section. Each approach has its own parameterisation, and we outline instructions for each below.

Random Walk and Linear Splines (Recommended)

For `looseformbeta` or `linearsplinebeta` analyses types, which we recommend using most of the time, you'll also need to provide an `initialBeta` parameter and `betaJitter`, which gives particles their initial beta value, and specifies the 'freedom' of the random walk (or slope of splines). Those will look like this:

```
      <initialBeta>
          <stepchange>false</stepchange>
          <disttype>Uniform</disttype>
          <min>0.05</min>
          <max>0.3</max>
      </initialBeta>
      <betaJitter>
          <stepchange>false</stepchange>
          <disttype>Uniform</disttype>
          <min>0.001</min>
          <max>0.05</max>
      </betaJitter>
```

Piecewise constant beta in epochs

If you wish to fit β in piecewise constant intervals, the approach is similar to any other parameter with a rate change. For example below, we provide a truncated normal prior for β with mean 0.6 during the initial growth phase of a theoretical epidemic (before day 50), followed by reducing this prior to a mean of 0.1.

```
<beta>
        <stepchange>true</stepchange>
        <changetime>
          <x0>
            <disttype>FixedParameter</disttype>
            <value>50</value>
          </x0>
        </changetime>
        <distribs>
          <x0>
            <disttype>TruncatedNormal</disttype>
            <mean>0.6</mean>
            <standarddev>0.1</standarddev>
            <lowerbound>0.0</lowerbound>
          </x0>
```

```
          <x1>
            <disttype>TruncatedNormal</disttype>
            <mean>0.1</mean>
            <standarddev>0.05</standarddev>
            <lowerbound>0.0</lowerbound>
          </x1>
        </distribs>
      </beta>
```

Inverse Logistic Function

If you are fitting beta as an inverse logistic function, it is parameterised a little differently (you provide priors for the three parameters of the inverse logistic curve:

```
      <a>
        <stepchange>false</stepchange>
          <disttype>Normal</disttype>
          <mean>0.047</mean>
          <standarddev>0.01</standarddev>
      </a>
      <b>
        <stepchange>false</stepchange>
        <disttype>Normal</disttype>
        <mean>-0.06</mean>
        <standarddev>0.01</standarddev>
      </b>
      <c>
        <stepchange>false</stepchange>
        <disttype>Normal</disttype>
        <mean>0.4</mean>
        <standarddev>0.1</standarddev>
      </c>
```

Appendix 6 - Time Variant Prior Distributions

In this article we include an example of using a 'time variant prior', i.e. allowing the some rates (e.g. case and genomic sequence sampling) to vary over time in piecewise constant intervals, with unique priors for each interval. Below we provide further information on this process, and expand on the options available for parameterising these changes.

Introducing the change

When you introduce a step-change by changing the parameter to `true`, you then need to set priors for each interval, and the interval change times. This moves `disttype` and the associated tags inside new XML nodes called `changetime` and `distribs`. The best way to explain this is with an example. First lets look at this parameterisation of psi, where psi is constant over time and has a truncated normal distribution:

```
      <psi>
          <stepchange>false</stepchange>
          <disttype>TruncatedNormal</disttype>
          <mean>0.00025</mean>
          <standarddev>0.00005</standarddev>
          <lowerbound>0.0</lowerbound>
      </psi>
```

Let's introduce a step-change where we expect psi to increase 10x at a specific time (day 35). First we set `stepchange` to `true`, then add `changetime` and `distribs` tags, inside which we set individual priors for the rates in each interval, and the interval times. These are now wrapped in tags `<x[n]>`, starting at `n=0` and counting up for every extra interval or rate you are adding. You can specify as many changetimes as you like, but you should have one more `distribs` element than the number of `changetime` elements.

```
<psi>
        <stepchange>true</stepchange>
        <changetime>
          <x0>
            <disttype>FixedParameter</disttype>
            <value>35</value>
          </x0>
        </changetime>
        <distribs>
          <x0>
            <disttype>TruncatedNormal</disttype>
            <mean>0.00025</mean>
            <standarddev>0.00005</standarddev>
            <lowerbound>0.0</lowerbound>
          </x0>
          <x1>
            <disttype>TruncatedNormal</disttype>
            <mean>0.0025</mean>
```

```
            <standarddev>0.0005</standarddev>
            <lowerbound>0.0</lowerbound>
          </x1>
        </distribs>
      </psi>
```

Inferring the changetime

Let's assume we don't know the time of the step-change, and we want to infer it. The result is the exact same as above, but `changetime <x0>` is now no longer a `FixedParameter`, but instead a `Poisson`.

```
<psi>
        <stepchange>true</stepchange>
        <changetime>
          <x0>
            <disttype>Poisson</disttype>
            <mean>35</mean>
          </x0>
        </changetime>
        <distribs>
          <x0>
            <disttype>TruncatedNormal</disttype>
            <mean>0.00025</mean>
            <standarddev>0.00005</standarddev>
            <lowerbound>0.0</lowerbound>
          </x0>
          <x1>
            <disttype>TruncatedNormal</disttype>
            <mean>0.0025</mean>
            <standarddev>0.0005</standarddev>
            <lowerbound>0.0</lowerbound>
          </x1>
        </distribs>
      </psi>
```

Introducing a 'buffer zone'

In our previous examples, the rate is parameterised to change instantaneously on the day of the 'changetime', and while this may be suitable for some examples, it is also common for these changes to take place more gradually, i.e. a more gradual increase in sampling as capacity scales. For this we provide the option to specify the `buffer` in the rate chunk:

```
<psi>
        <stepchange>true</stepchange>
        <buffer>10</buffer>
        <changetime>
          <x0>
            <disttype>Poisson</disttype>
```

```
            <mean>35</mean>
          </x0>
        </changetime>
        <distribs>
          <x0>
            <disttype>TruncatedNormal</disttype>
            <mean>0.00025</mean>
            <standarddev>0.00005</standarddev>
            <lowerbound>0.0</lowerbound>
          </x0>
          <x1>
            <disttype>TruncatedNormal</disttype>
            <mean>0.0025</mean>
            <standarddev>0.0005</standarddev>
            <lowerbound>0.0</lowerbound>
          </x1>
        </distribs>
      </psi>
```

Here we provide a buffer of 10, meaning that the model will sample rates for each segment, begin to linearly adjust the rates between segments from 10 days before to 10 days after the `changetime`. This makes the change more gradual, and the resulting rate trajectories, including the effect of the buffer, will be saved in .csv files from the analysis in the output folder.