# Model selection with nested sampling

## Tutorial using BEAST v2.5.2

*Remco Bouckaert*

## 1 Background

BEAST provides a bewildering number of models. Bayesians have two techniques to deal with this problem: model averaging and model selection. With **model averaging**, the MCMC algorithms jumps between the different models, and models more appropriate for the data will be sampled more often than unsuitable models (see for example the substitution model averaging tutorial). **Model selection** on the other hand just picks one model and uses only that model during the MCMC run. This tutorial gives some guidelines on how to select the model that is most appropriate for your analysis.

Bayesian model selection is based on estimating the marginal likelihood: the term forming the denominator in Bayes formula. This is generally a computationally intensive task and there are several ways to estimate them. Here, we concentrate on nested sampling as a way to estimate the marginal likelihood as well as the uncertainty in that estimate.

Say, we have two models, M1 and M2, and estimates of the (log) marginal likelihood, ML1 and ML2, then we can calculate the Bayes factor, which is the fraction BF=ML1/ML2 (or in log space, the difference log(BF) = log(ML1)-log(ML2)). If BF is larger than 1, model M1 is favoured, and otherwise M2 is favoured. How much it is favoured can be found in the following table (Kass and Raftery 1995):

| $BF$ range | $\ln(BF)$ range | $\log_{10}(BF)$ range | Interpretation |
|---|---|---|---|
| 1 – 3 | 0 – 1.1 | 0 – 0.5 | hardly worth mentioning |
| 3 – 20 | 1.1 – 3 | 0.5 – 1.3 | positive support |
| 20 – 150 | 3 – 5 | 1.3 – 2.2 | strong support |
| > 150 | > 5 | > 2.2 | overwhelming support |

Figure 1: Bayes factor support.

Note that sometimes a factor 2 is used for multiplying BFs, so when comparing BFs from different publications, be aware which definition was used.

**Nested sampling** is an algorithm that works as follows:

- randomly sample `N` points from the prior
- while not converged
  - pick the point with the lowest likelihood $L_{min}$, and save to log file
  - replace the point with a new point randomly sampled from the prior using an MCMC chain of `subChainLength` samples **under the condition that the likelihood is at least** $L_{min}$

So, the main parameters of the algorithm are the number of particles `N` and the `subChainLength`. `N` can be determined by starting with `N=1` and from the information of that run a target standard deviation can be determined, which gives us a formula to determine `N` (as we will see later in the tutorial). The `subChainLength` determines how independent the replacement point is from the point that was saved, and is the only parameter that needs to be determined by trial and error – see FAQ for details.

# 2 Programs used in this tutorial

### 2.0.1 BEAST2 - Bayesian Evolutionary Analysis Sampling Trees 2

BEAST2 is a free software package for Bayesian evolutionary analysis of molecular sequences using MCMC and strictly oriented toward inference using rooted, time-measured phylogenetic trees (Bouckaert et al. 2014), (Bouckaert et al. 2018). This tutorial uses the BEAST2 version 2.5.2.

### 2.0.2 BEAUti2 - Bayesian Evolutionary Analysis Utility

BEAUti2 is a graphical user interface tool for generating BEAST2 XML configuration files.

Both BEAST2 and BEAUti2 are Java programs, which means that the exact same code runs on all platforms. For us it simply means that the interface will be the same on all platforms. The screenshots used in this tutorial are taken on a Mac OS X computer; however, both programs will have the same layout and functionality on both Windows and Linux. BEAUti2 is provided as a part of the BEAST2 package so you do not need to install it separately.

### 2.0.3 Tracer

Tracer is used to summarise the posterior estimates of the various parameters sampled by the Markov Chain. This program can be used for visual inspection and to assess convergence. It helps to quickly view median estimates and 95% highest posterior density intervals of the parameters, and calculates the effective sample sizes (ESS) of parameters. It can also be used to investigate potential parameter correlations. We will be using Tracer v1.7.0.

# 3 Practical: Selecting a clock model

We will analyse a set of hepatitis B virus (HBV) sequences sampled through time and concentrate on selecting a clock model. The most popular clock models are the strict clock model and uncorrelated relaxed clock with log normal distributed rates (UCLN) model.

## 3.1 Setting up the Strict clock analysis

First thing to do is set up the two analyses in BEAUti, and run them in order to make sure there are differences in the analyses. The alignment can be downloaded here: https://raw.githubusercontent.com/rbouckaert/NS-tutorial/master/data/HBV.nex. We will set up a model with tip dates, HKY substitution model, Coalescent prior with constant population, and a fixed clock rate.

**In BEAUti:**

- Start a new analysis using the Standard template.
- Import HBV.nex using menu `File` > `Import alignment`.
- In the tip-dates panel, select `tip dates`, click `Auto configure` and select the `split on character` option, taking group 2 (see Fig 2).
- In the site model panel, select `HKY` as substitution model and leave the rest as is.
- In the clock model panel, set the clock rate to 2e-5. Though usually, we want to estimate the rate, to speed things up for the tutorial, we fix the clock rate at that number as follows:
  - Uncheck menu `Mode` > `Automatic set clock rate`. Now the estimate entry should not be grayed out any more.
  - Uncheck the `estimate` box next to the clock rate entry.
- In the priors panel, select `Coalescent Constant Population` as tree prior.
- Also in the priors panel, change to `popSize` prior to `Gamma` with alpha = 0.01, beta = 100 (Fig 3).
- In the MCMC panel, change the `Chain Length` to 1 million.
- You can rename the file for trace log and tree file to include "Strict" to distinguish them for the relaxed clock ones.
- Save the file as `HBVStrict.xml`. **Do not close BEAUti just yet!**
- Run the analysis with BEAST.

**Do you have a clock rate prior in the priors panel?** If so, the clock rate is estimated, and you should revisit the part where the clock is set up!

## 3.2 Setting up the relaxed clock analysis

While you are waiting for BEAST to finish, it is time to set up the relaxed clock analysis. This is now straightforward if BEAUti is still open (if BEAUti was closed, open BEAUti, and load the file `HBVStrict.xml` through the menu `File` > `Load`):

- In the clock model panel, change `Strict clock` to `Relaxed Clock Log Normal`.
- Set the clock rate to 2e-5, and uncheck the `estimate` box.
- In the MCMC panel, replace `Strict` in the file names for trace and tree log to `UCLN`.

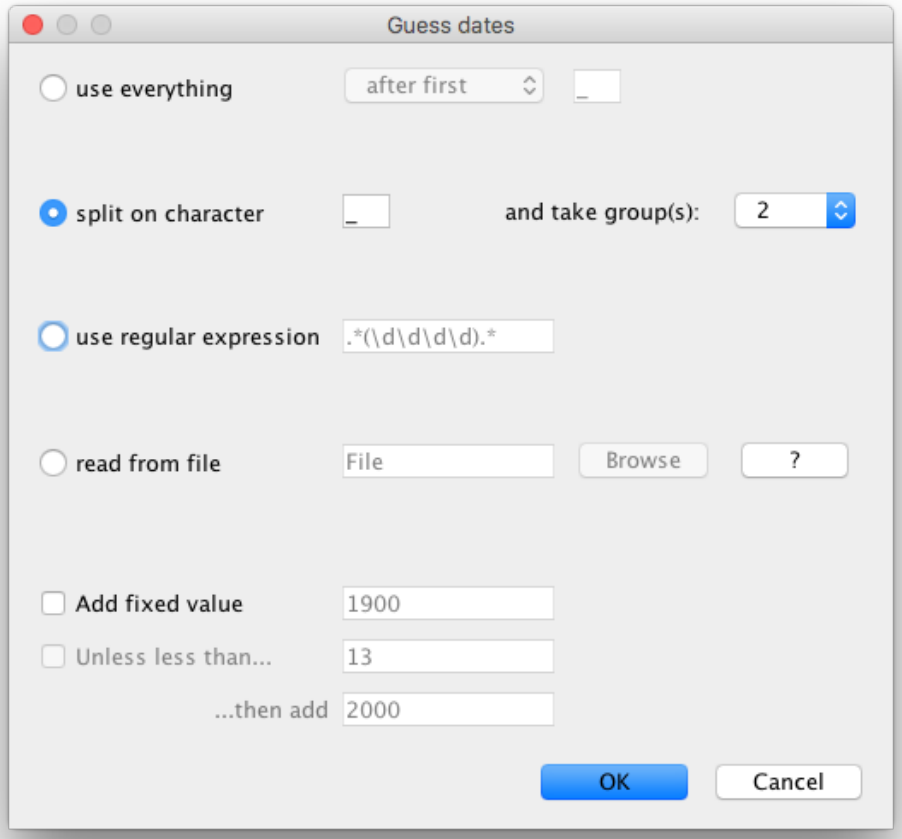


Figure 2: Configuring tip dates in BEAUti

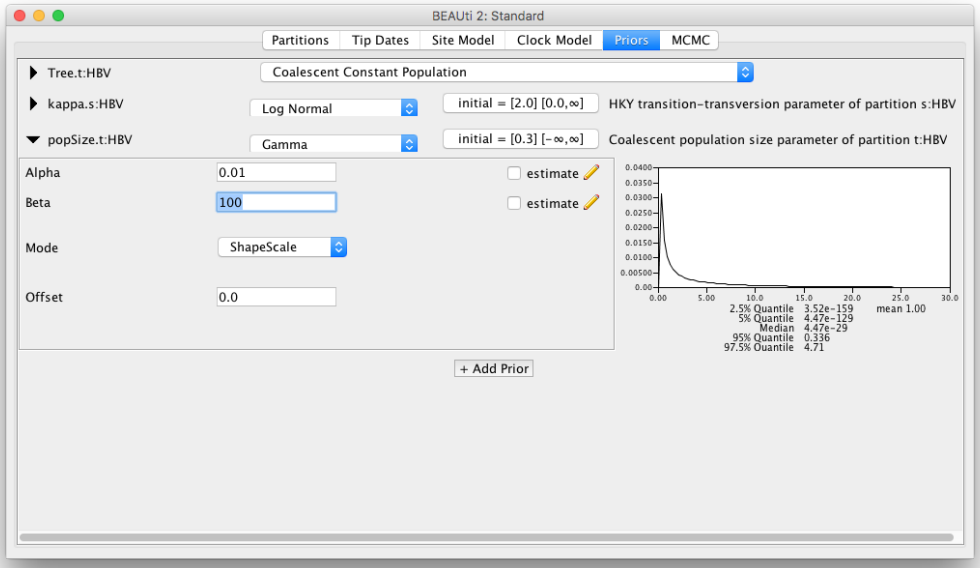


Figure 3: Priors panel for strict clock analysis in BEAUti

- Save file as `HBVUCLN.xml` **Do not click the** `File` > `Save` **menu, but** `File` > `Save as`**, otherwise the strict clock XML file will be overwritten**
- Run the analysis in BEAST

Once the analyses have run, open the log file in Tracer and compare estimates and see whether the analyses substantially differ. You can also compare the trees in DensiTree.

Are there any statistics that are very different? Do tree heights differ substantially? Which analysis is preferable and why?

If there are no substantial differences between the analysis for the question you are interested in, you do not have to commit to one model or another, and you can claim that the results are robust under different models. However, if there are significant differences, you may want to do a formal test to see which model is preferred over other models. In a Bayesian context, in practice this comes down to estimating the marginal likelihood, and calculating Bayes factors: the ratios of marginal likelihoods. Nested sampling (Russel et al. 2018) is one way to estimate marginal likelihoods.

## 3.3 Installing the NS Package

To use nested sampling, first have to install the NS (version {{ page.nsversion }} or above) package.

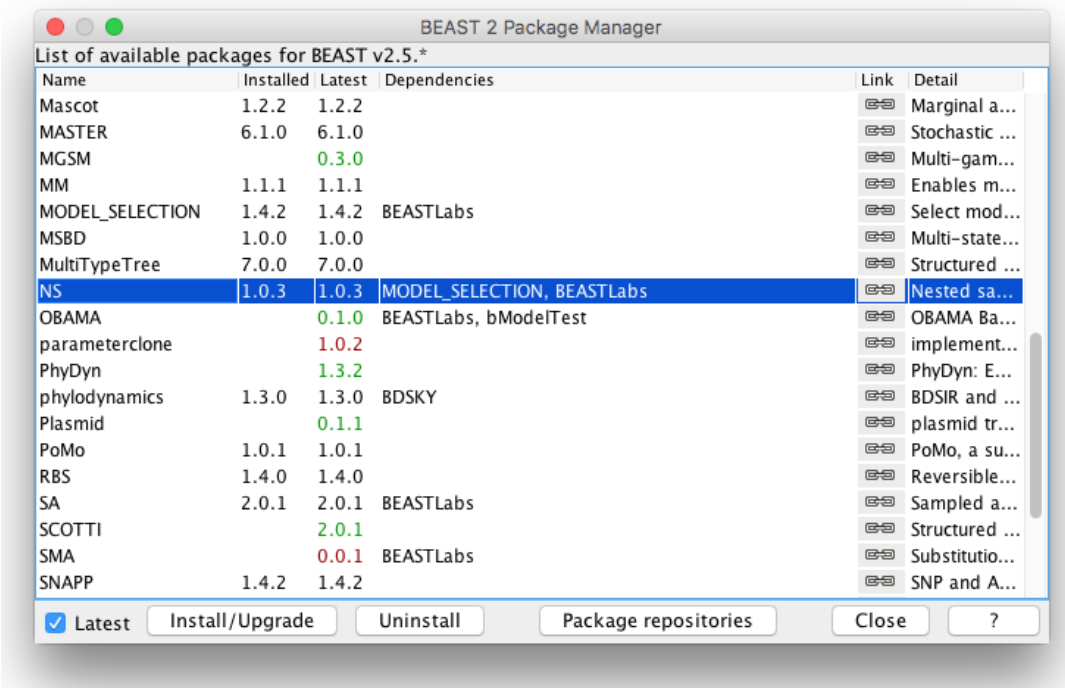


Figure 4: Installing NS in the Manage Packages window in BEAUti

Open BEAUti and navigate to **File > Manage Packages**. Select NS and then click **Install/Upgrade** (Fig 4). Then ***restart BEAUti*** to load the package.

## 3.4 Setting up the nested sampling analyses

- copy the file `HBVStrict.xml` to `HBVStric-NS.xml` and
- copy `HBVUCLN.xml` to `HBVUCLN-NS.xml`
- start a text editor and in both copied files, change

```
<run id="mcmc" spec="MCMC" chainLength="1000000">
```

to

```
<run id="mcmc" spec="beast.gss.NS" chainLength="20000" particleCount="1" subChainLength="5000↩
    ">
```

- Here the `particleCount` represents the number of active points used in nested sampling: the more points used, the more accurate the estimate, but the longer the analysis takes. The `subChainLength` is the number of MCMC samples taken to get a new point that is independent (enough) from the point that is saved. Longer lengths mean longer runs, but also more independent samples.

In practice, running with different `subChainLength` is necessary to find out which length is most suitable (see FAQ).

- change the file names for the trace and tree log to include `NS` (searching for `fileName=` will get you there fastest).
- save the files, and run with BEAST.

The end of the BEAST run for nested sampling with the strict clock should look something like this:

```
Total calculation time: 34.146 seconds
End likelihood: -202.93224422946253
Producing posterior samples

Marginal likelihood: -12438.35758179847 sqrt(H/N)=(11.084732655710818)=?=SD=(11.008249475863373) ↩
    Information: 122.87129804858182
Max ESS: 6.706301939836118


Processing 248 trees from file.
Log file written to HBVStrict-NS.posterior.trees
Done!

Marginal likelihood: -12437.767705819117 sqrt(H/N)=(11.05930077842042)=?=SD=(11.71074051847201) ↩
    Information: 122.3081337075705
Max ESS: 6.490414156277384


Log file written to HBVStrict-NS.posterior.log
Done!
```

and for the relaxed clock:

```
Total calculation time: 38.541 seconds
End likelihood: -200.7794173971489
Producing posterior samples

Marginal likelihood: -12428.557546706481 sqrt(H/N)=(11.22272275528845)=?=SD=(11.252847709777592) ↩
    Information: 125.94950604206919
Max ESS: 5.874085822198268


Processing 257 trees from file.
Log file written to HBVUCLN-NS.posterior.trees
Done!

Marginal likelihood: -12428.480923049345 sqrt(H/N)=(11.220392192278625)=?=SD=(11.491864352217954)↩
     Information: 125.89720094854714
Max ESS: 5.940996269769591


Log file written to HBVUCLN-NS.posterior.log
Done!
```

As you can see, nested sampling produces estimates of the marginal likelihood as well as standard deviation estimates. At first sight, the relaxed clock has a log marginal likelihood estimate of about -12428, while the strict clock is much worse at about -12438. However, the standard deviation of both runs is about 11, so that makes these estimates indistinguishable. Since this is a stochastic process, the exact numbers for

your run will differ, but should not be that far appart (less than 2 SDs, or about 22 log points in 95% of the time).

To get more accurate estimates, the number of particles can be increased. The expected SD is sqrt(H/N) where N is the number of particles and H the information. The information H is conveniently estimated in the nested sampling run as well. To aim for an SD of say 2, we need to run again with N particles such that 2=sqrt(125/N), which means 4=125/N, so N=125/4 and N=32 will do. Note that the computation time of nested sampling is linear in the number of particles, so it will take about 32 times longer to run if we change the particleCount from 1 to 32 in the XML.

A pre-cooked run with 32 particles can be found here: https://github.com/rbouckaert/NS-tutorial/tree/master/precooked_runs. Download the files HBV-Strict-NS32.log and HBVUCLN-NS32.log and run the `NSLogAnalyser` application to analyse the results. To start the `NSLogAnalyser` from the command line, use

```
applauncher NSLogAnalyser -noposterior -N 32 -log /path/to/HBVStrict-NS32.log
applauncher NSLogAnalyser -noposterior -N 32 -log /path/to/HBVUCLN-NS32.log
```

or from BEAUti, click menu `File/Launch apps`, select `NSLogAnalyser` and fill in the form in the GUI. The output for the strict clock analysis should be something like this:

```
Loading HBVStrict-NS32.log, burnin 0%, skipping 0 log lines

|---------|---------|---------|---------|---------|---------|---------|---------|
****************************

Marginal likelihood: -12426.207750474812 sqrt(H/N)=(1.8913059067381148)=?=SD=(1.8374367294317693)↩
     Information: 114.46521705159945
Max ESS: 400.41214209052896

Calculating statistics

|---------|---------|---------|---------|---------|---------|---------|---------|
*******************************************************************************

#Particles = 32
item                mean      stddev
posterior           -12512.7  2.988107
likelihood          -12311.7  2.91633
prior               -201.009  1.580207
treeLikelihood      -12311.7  2.91633
TreeHeight          3443.955  134.1921
kappa               2.679169  0.151243
popSize             2337.454  290.0407
CoalescentConstant  -163.191  3.090671
freqParameter.1     0.240033  0.005884
freqParameter.2     0.267856  0.006814
freqParameter.3     0.217084  0.006638
freqParameter.4     0.275027  0.006716
Done!
```

So, that gives us a ML estimate of -12426.2 with SD of 1.89, slightly better than the 2 we aimed for, but the information is also a bit lower than we assumed (114 vs 128). Furthermore, there are posterior estimates of all the entries in the trace log. Nested sampling does not only estimate MLs and SDs, but can also provide a sample from the posterior, which can be useful for cases where MCMC has trouble with convergence. But let's not digress too much and get back to model selection.

For the relaxed clock analysis, we get something like:

```
Marginal likelihood: -12417.389793288146 sqrt(H/N)=(1.9543337689486355)=?=SD=(1.9614418034828585)↩
     Information: 122.2214553744953
```

so an ML of -12417.4 with SD of 1.95. Therefor the log BF is -12417.4 - -12426.2 = 8.8, which is more than twice the sum of the SDs, so can be considered reliable evidence in favour of the relaxed clock model. Note that judging from the table at the start of the tutorial, this amounts to overwhelming support for the relaxed clock.

# 4 Nested sampling FAQ

## 4.1 The analysis prints out multiple ML estimates with their SDs. Which one to choose?

The difference between the estimates is the way they are estimated from the nested sampling run. Since these are estimates that require random sampling, they differ from one estimate to another. When the standard deviation is small, the estimates will be very close, but when the standard deviations is quite large, the ML estimates can substantially differ. Regardless, any of the reported estimates are valid estimates, but make sure to report them with their standard deviation.

## 4.2 How do I know the sub-chain length is large enough?

NS works in theory if and only if the points generated at each iteration are independent. If you already did an MCMC run and know the effective sample size (ESS) for each parameter, to be sure every parameter in every sample is independent you can take the length of the MCMC run divided by the smallest ESS as sub-chain length. This tend to result in quite large sub-chain lengths.

In practice, we can get away much smaller sub-chain lengths, which you can verify by running multiple NS analysis with increasing sub-chain lengths. If the ML and SD estimates do not substantially differ, you know the shorter sub-chain length was sufficient.

## 4.3 How many particles do I need?

To start, use only a few particles. This should give you a sense of the information `H`, which is one of the estimates provided by the NS analysis. If you want to compare two hypotheses, you want the difference between `ML1` and `ML2` to be at least `2*sqrt(SD1*SD1+SD2*SD2)` in order to make sure the difference is not due to randomisation.

If the difference is larger, you do not need more particles.

If the difference is smaller, you can guess how much the SD estimates must shrink to get a difference that is sufficiently large. Since the `SD=sqrt(H/N)`, we have that `N=H/(SD*SD)` and `H` comes from the NS run with a few particles. Run the analysis again, with the increased number of particles, and see if the difference becomes large enough.

If the difference is less than 2, the hypotheses may not be distinguishable – in terms of Bayes factors, are barely worth mentioning.

### 4.4 Is NS faster than path sampling/stepping stone (PS/SS)?

This depends on many things, but in general, depends on how accurate the estimates should be. For NS, we get an estimate of the SD, which is not available for PS/SS. If the hypotheses have very large differences in MLs, NS requires very few (maybe just 1) particle, and will be very fast. If differences are smaller, more particles may be required, and the run-time of NS is linear in the number of particles.

The parallel implementation makes it possible to run many particles in parallel, giving a many-particle estimate in the same time as a single particle estimate (PS/SS can be parallelised by steps as well).

### 4.5 The output is written on screen, which I forgot to save. Can I estimate them directly from the log files?

The NS package has a `NSLogAnalyser` application that you can run via the menu `File/Launch apps` in BEAUti – a window pops up where you select the `NSLogAnalyser`, and a dialog shows you various options to fill in. You can also run it from the command line on OS X or Linux using

```
/path/to/beast/bin/applauncher NSLogAnalyser -N 1 -log  xyz.log
```

where the argument after `N` is the `particleCount` you specified in the XML, and `xyz.log` the trace log produced by the NS run.

### 4.6 Why are some NS runs longer than others?

Nested sampling stops automatically when the accuracy in the ML estimate cannot be improved upon. Because it is a stochastic process, some analyses get there faster than others, resulting in different run times.

### 4.7 Why are the ESSs so low when I open a log file in Tracer?

An NS analysis produces two trace log files: one for the nested sampling run (say `myFile.log`) and one with the posterior sample (`myFile.posterior.log`).

The ESSs in Tracer of log files with the posterior samples are meaningless, because the log file is ordered using the nested sampling run. If you look at the trace of the Likelihood, it should show a continuous increasing function. It is not quite clear how to estimate ESSs of a nested sampling run yet, though the number of entries in the posterior log is equal to the maximum theoretical ESS, which is almost surely an overestimate.

# 5 Useful Links

- Bayesian Evolutionary Analysis with BEAST 2 (Drummond and Bouckaert 2014)
- BEAST 2 website and documentation: http://www.beast2.org/
- Nested sampling website and documentation: https://github.com/BEAST2-Dev/nested-sampling
- Join the BEAST user discussion: http://groups.google.com/group/beast-users

This tutorial was written by Remco Bouckaert for Taming the BEAST and is licensed under a Creative Commons Attribution 4.0 International License.

Version dated: November 19, 2020

# Relevant References

Bouckaert, R, J Heled, D Kühnert, T Vaughan, CH Wu, D Xie, MA Suchard, A Rambaut, and AJ Drummond. 2014. Beast 2: a software platform for bayesian evolutionary analysis. *PLoS computational biology* 10: e1003537.

Bouckaert, R, TG Vaughan, J Barido-Sottani, S Duchene, M Fourment, A Gavryushkina, J Heled, G Jones, D Kuhnert, N De Maio, et al. 2018. Beast 2.5: an advanced software platform for bayesian evolutionary analysis. *BioRxiv* 474296.

Drummond, AJ and RR Bouckaert. 2014. *Bayesian evolutionary analysis with BEAST 2*. Cambridge University Press,

Kass, RE and AE Raftery. 1995. Bayes factors. *Journal of the american statistical association* 90: 773–795.

Russel, PM, BJ Brewer, S Klaere, and RR Bouckaert. 2018. Model selection and parameter inference in phylogenetics using nested sampling. *Systematic biology* 68: 219–233.