BEAST lab

Genomics of Disease in Wildlife Workshop
June 8, 2023

Activities:

1. We will perform a discrete trait analysis with a skyride model on equine+canine H3N8. This activity will likely take the entire. I've also included a 2nd activity with the resulting output files for you to explore, but I think it is unlikely that we will get to it in this lab.
2. Bonus activity: We will perform a local clock analysis to compare whether there is a difference in evolutionary rate between these two parts of the phylogeny

Outline/background:

This activity/lab builds on the phylogenies we built in Nextstrain. In the 1960s, an H3N8 influenza virus was detected in horses in Florida. This lineage then transmitted between horses, resulting in a spillover into dogs. This lineage circulated in dogs for several years before dying out recently. In this tutorial, we will learn how to use BEAST to infer when H3N8 was transmitted from horses to dogs, how rapidly the virus population grew during transmission in horses and dogs, and measure whether the receptor binding protein (HA) is evolving at different rates in horses and dogs. We will evaluate run convergence and mixing using Tracer, plot posterior distributions, and generate a summary tree.

For both analyses in this lab, we will be using BEAST v.1.10. BEAST 1 and BEAST 2 are actually quite different pieces of software, and include different arrays of models. BEAST is developed by a large group of highly collaborative investigators, who specialize in different types of models. These investigators often stick with development in either BEAST 1 or BEAST 2, and as a result, the interaction interface and organization, is somewhat distinct between the two. For discrete trait analyses (phylogeography), I usually use BEAST 1. For anything involving structured coalescent models or birth-death models, I always use BEAST 2. Both BEASTs include a complementary package called Beauti that acts as a user interface to make BEAST input files. We will be using Beauti to generate our input files, which we will then run in BEAST. We will use Tracer to evaluate chain convergence, and TreeAnnotator to generate summary trees.

Input data:

The input data for both analyses is an alignment of publicly available influenza HA gene segments sampled globally. These data have been organized with metadata describing the date of sample collection, subtype, geographic region and country of sampling, and host group. These sequences were then aligned with MAFFT.

1. First, take a look at the input alignment. It is always a good idea to look at your alignment before starting a BEAST analysis. If there are issues with your alignment, you will likely have problems with BEAST as well. To take a look, open SeaView, and drag and drop your aligned fasta file, `h3nx_canine_equine_ha.fasta` into it.

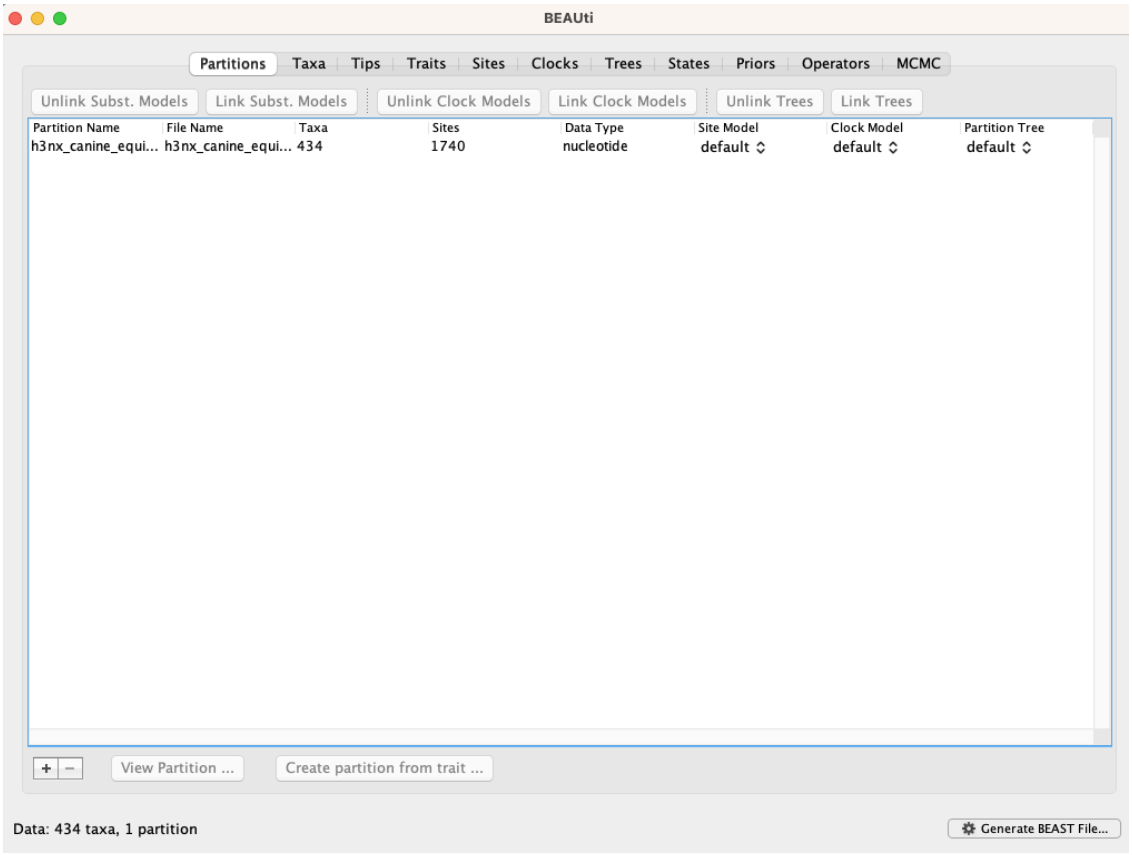
Analysis 1: Discrete traits + skyline model

In this analysis, we will estimate a time-resolved phylogeny using a skyride coalescent model. This model allows the population size to change over time, and both infers the appropriate number of intervals over which the population size has changed, and infers the population size across those intervals. The number of intervals is set to the number of taxa, such that there will be that many number of coalescent intervals for all of the taxa to coalesce. The model will use the time that lineages take to coalesce to estimate these population sizes, which we will summarize and can interpret as the degree of viral transmission intensity over time. In addition to estimating population size changes over time, we will also perform two reconstructions on the tree using a

“discrete trait” approach. This approach is analogous to the reconstructions we performed in the Nextstrain lab to color internal nodes by host and geographic regions. We will perform the same analysis now in BEAST, and compare results. BEAST will also output an estimate of the frequency of transmission events between groups, ie., hosts or geographic regions. We will interpret these results as the rate of transmission between geographic regions, and between hosts. Finally, by visualizing these results on a phylogenetic tree, we can infer the timing of these events, i.e., the date at which H3N8 was transmitted from horses to dogs.

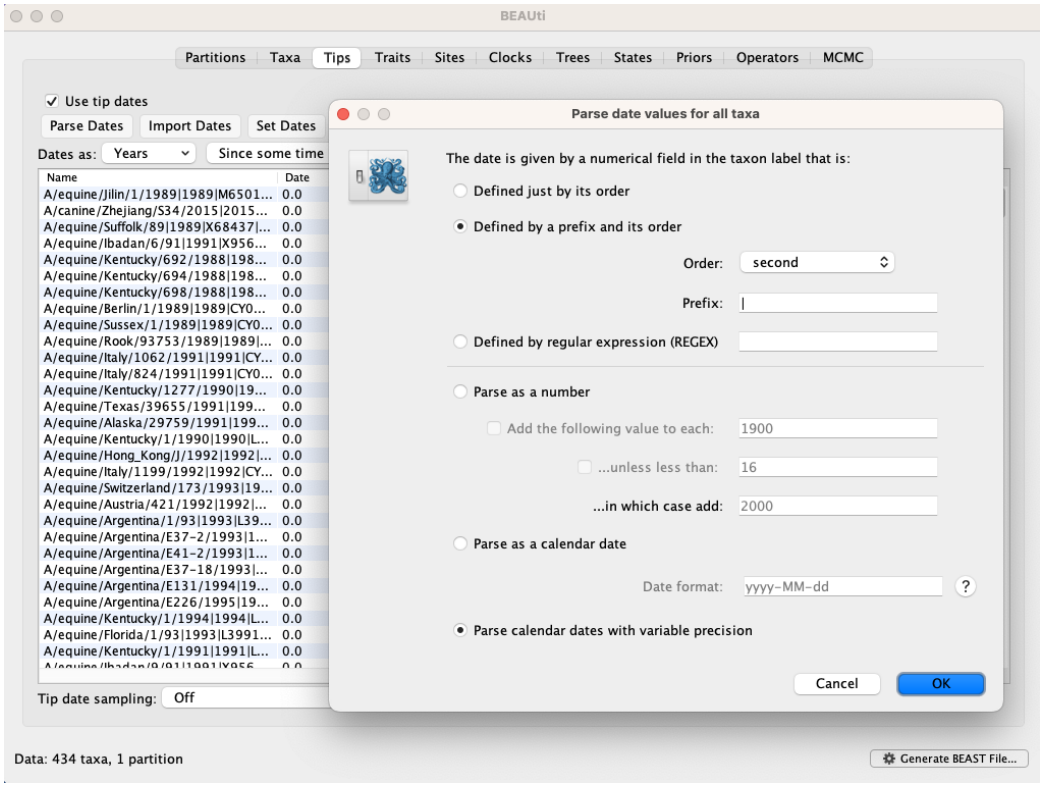
Part 1: Setting up the analysis

1. To get started, open Beauti by double clicking on it. To load your alignment into Beauti, drag your xml onto the main “Partitions” screen. It should load the alignment, and look like this:

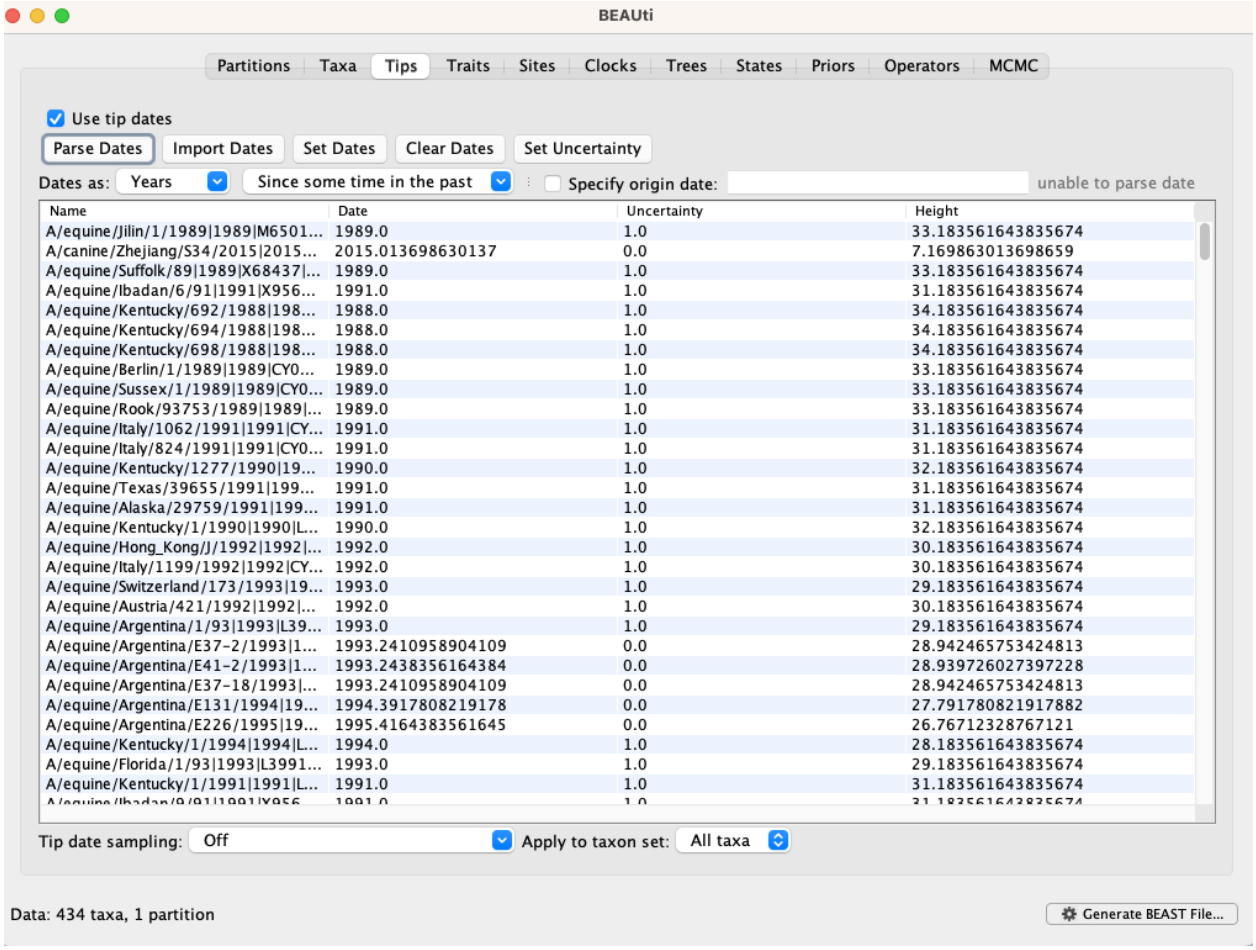


You can see here that there are 434 sequences in this alignment (here, called “taxa”), and that the alignment is made of nucleotides.

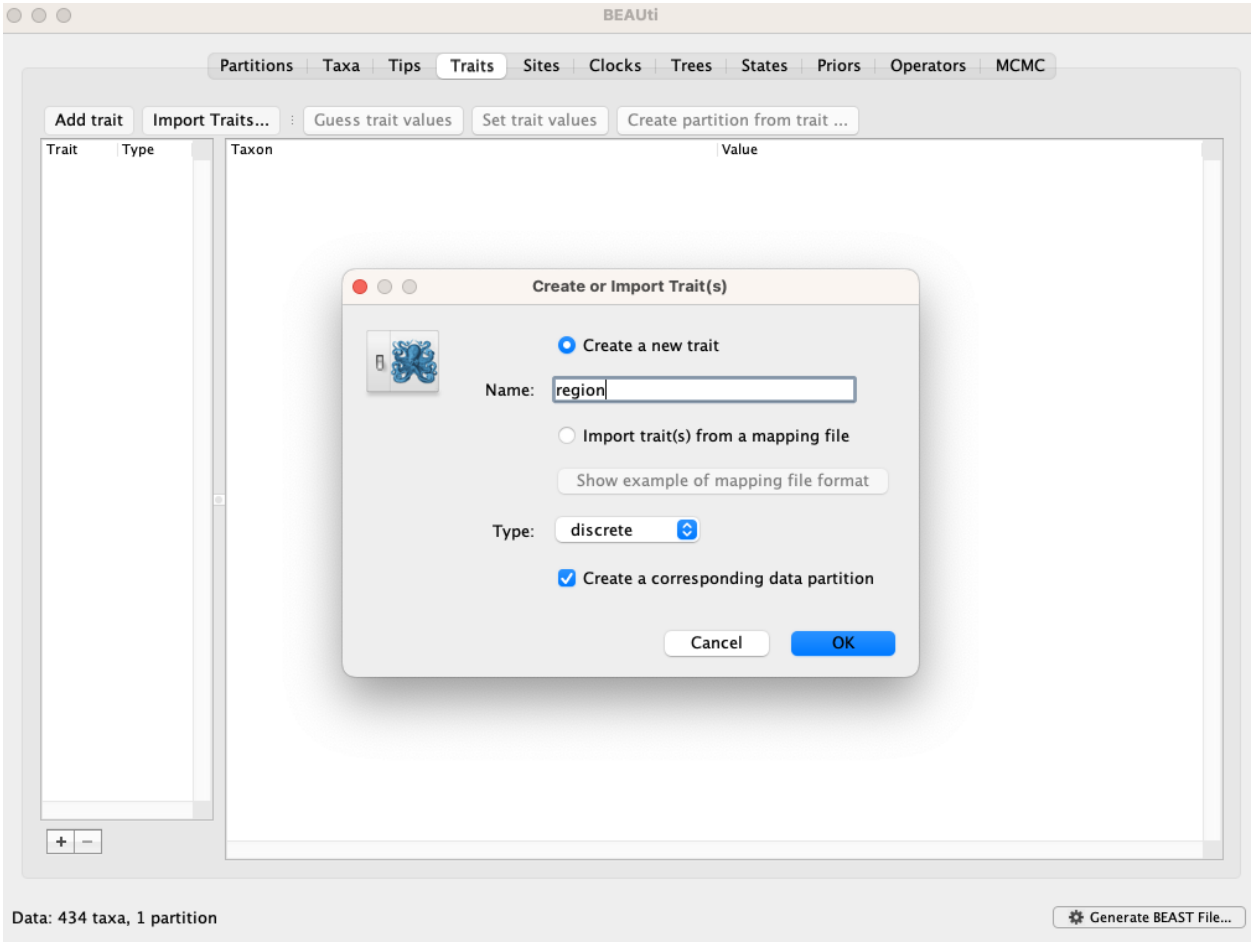
2. Next, we will specify our tip dates. Click on the “Tips” tab and and select “use tip dates.” BEAST allows you to either import tip dates separately, or to “guess” dates from information in your sequence headers. Our data has tip dates encoded in the sequence headers, so we can select that option. Select “Parse Dates”, and in the new pop-up window, select the options shown below. Parsing calendar dates with variable precision tells BEAST that we have some dates for which we have complete collection dates (including, year, month, and day), and others for which we only have a subset of information (usually just year, or year and month). Clicking this option tells BEAST to interpret the dates that are incomplete as less precise than complete dates.



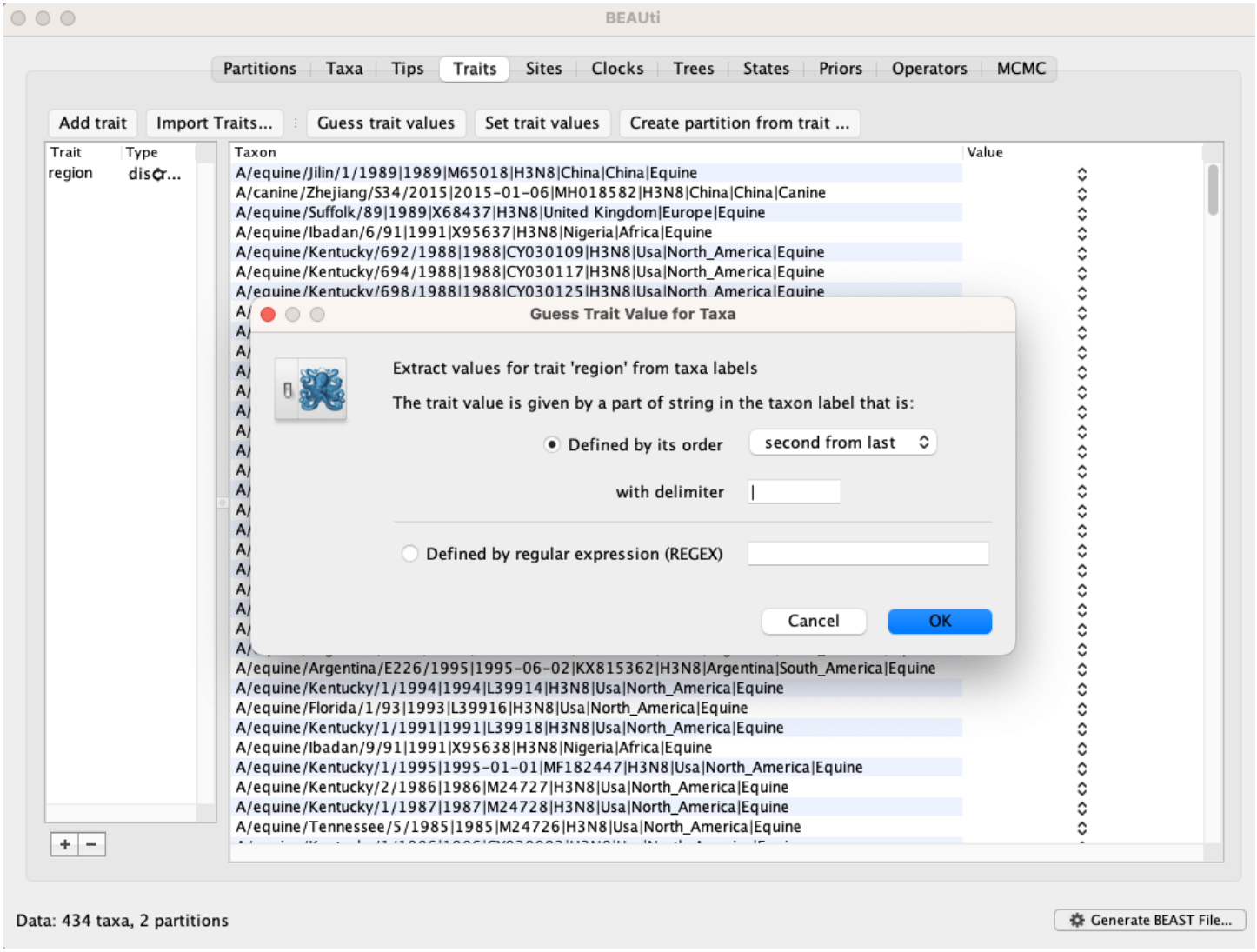
Click on “Ok”, and the dates should now be auto-filled in the “Date” column. Notice that for samples with complete collection dates, the Uncertainty is set to 0.0, whereas samples with year-only collection dates have Uncertainty of 1.0, meaning 1 full year of uncertainty.



3. Next, we will set up our discrete trait analysis. We will add 2 traits: region, and host. Click on “Add Trait”, and name it “region”. Keep the type as “discrete” and keep the “Create a corresponding data partition” checked. It should look like this:

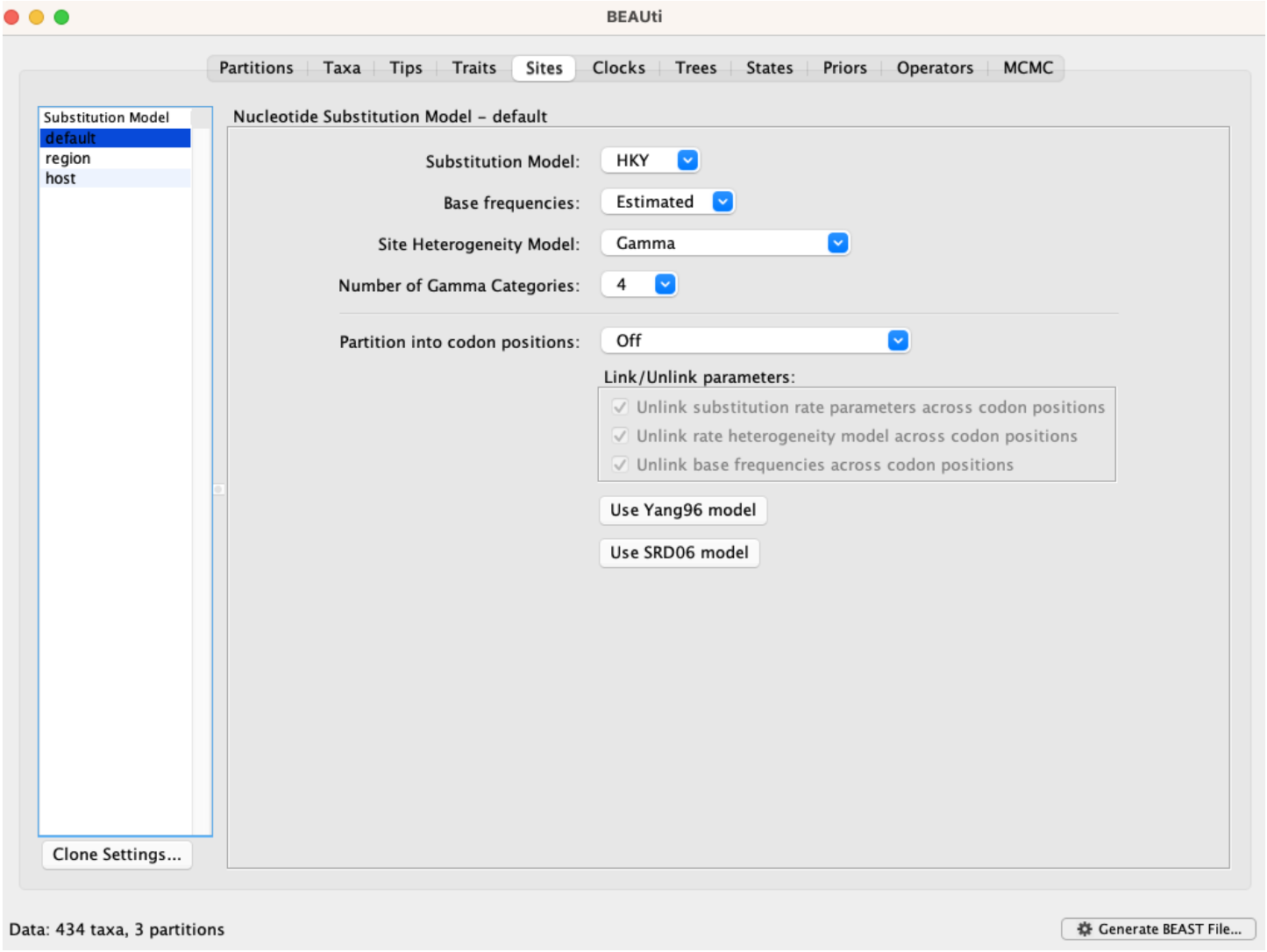


Now, to specify those values, click on “Guess Trait values” and set the region trait to second from last, with the “|” as delimiter. Click “OK”, and the regions should auto-fill.



Repeat the same process to make a trait called “host”, which is specific as the last partition with the “|” delimiter.

4. Now, we will set our nucleotide substitution model. For this analysis, select an HKY model, with estimated base frequencies, and a Gamma site heterogeneity model with 4 categories.

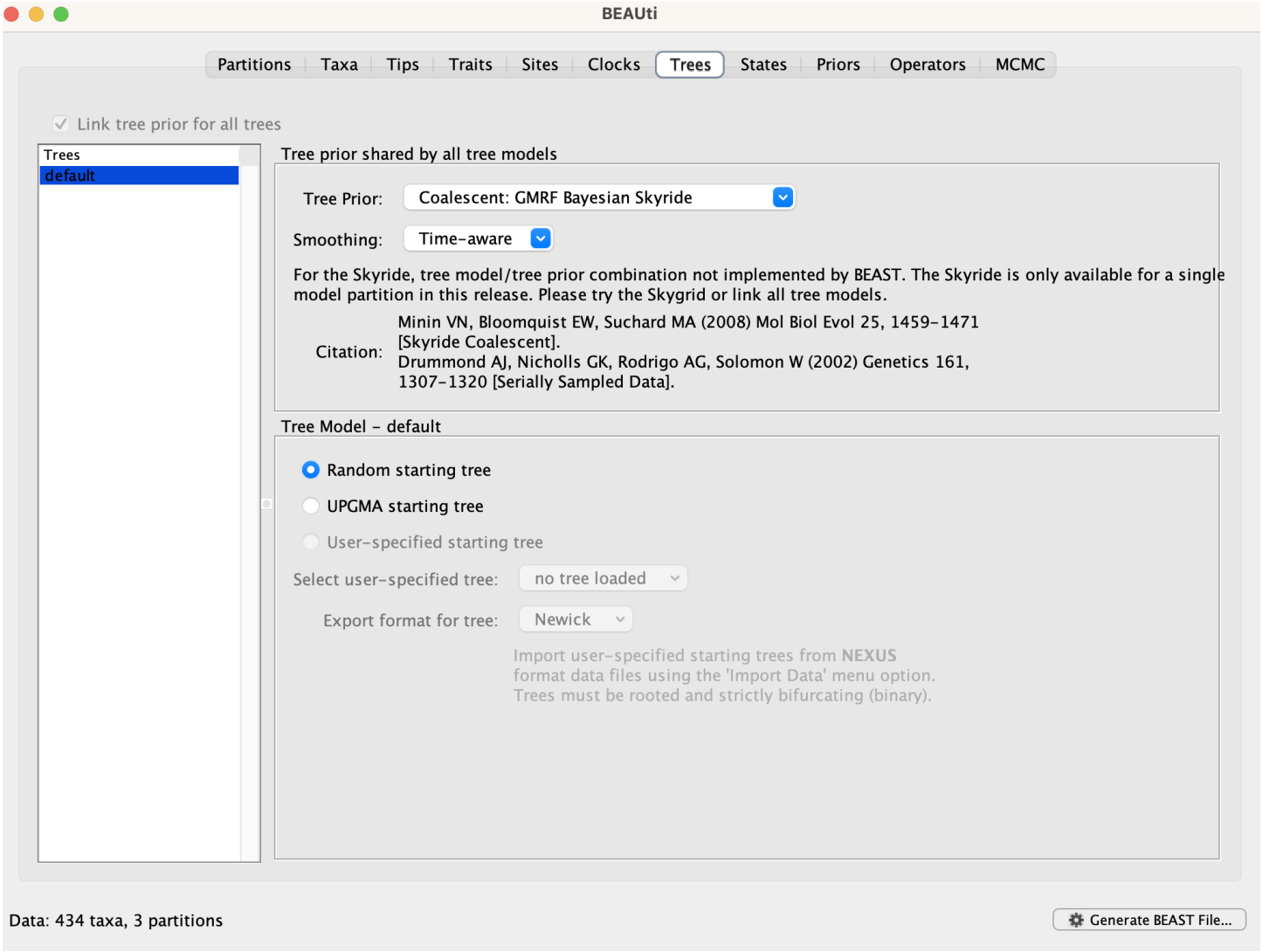


We also need to set a “substitution model” for our discrete trait analysis. This corresponds to the rate of transition between our categories (e.g., North America to South America or canine to equine). On the left side of the screen, click on “region” and select “symmetric substitution model” and “infer social network with BSSVS”. The symmetric substitution model means that the rate of transmission between regions is the same. For example, the inferred transmission rate from North America to South America is equal to South America to North America, so we will only infer 1 rate per pair of locations. Although this may not be realistic, for a first pass, this will greatly reduce the number of parameters we need to estimate. BSSVS is a method for reducing computational complexity by allowing transmission rates to be turned fully on and off. It essentially adds 0/1 indicator variables to each estimated rate, allowing the model to only infer the rates that matter. I don’t know why it is called “inferring a social network”, but BSSVS is useful and I generally always use it.

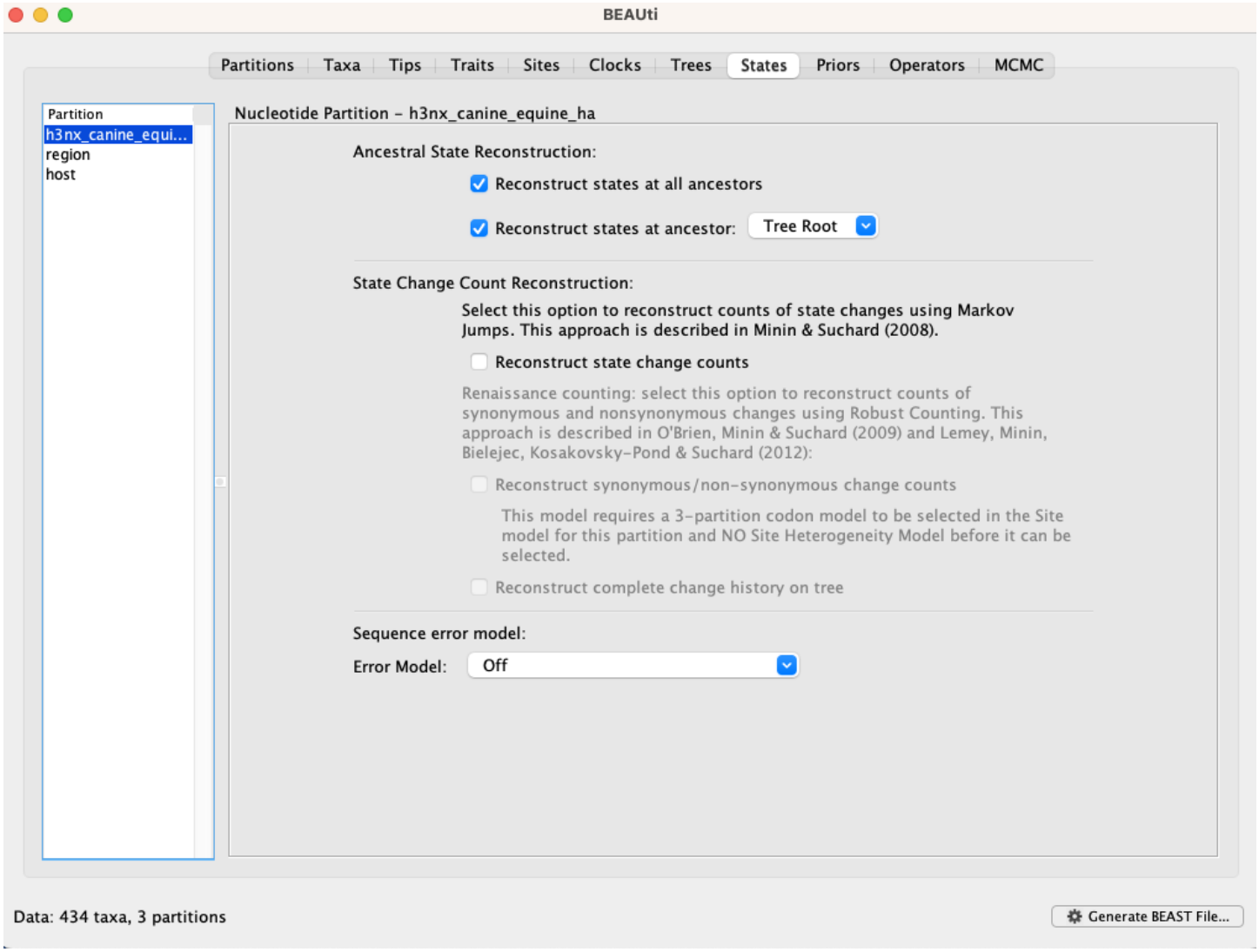
Repeat the same process for the host category, but set the transmission rates to be asymmetrical. For both traits, you’ll see an option for “set up a GLM”. A GLM, or generalized linear model is a cool way to extend these analyses to incorporate other information into a phylodynamic reconstruction. However, they are beyond the scope of this tutorial.

5. Setting clocks. People feel very specifically about clocks and have their own preferences. We will be investigating using alternative, host-specific clocks in the 2nd activity. For this, keep a strict clock for all categories.

6. In the Trees section, we specify what type of model we want for our phylogeny. We would like to investigate how the population size of H3N8 influenza changed during it's introduction and spread in horses and dogs. To do so, we will be using a skyride model. The skyride model attempts to estimate the appropriate number of bins that are necessary to model the changes in population size over time.

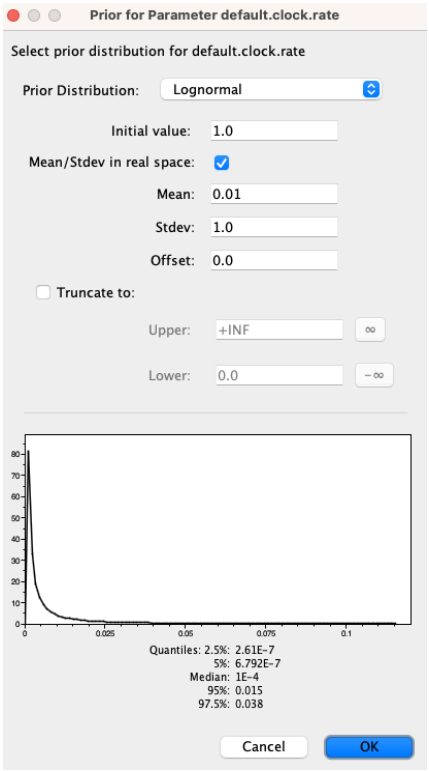


7. States. This specifies whether we would like to reconstruct ancestral states at internal nodes. This usually doesn't add significant computational time, and is sometimes fun and useful. For the nucleotide partition, this simply means inferring nucleotide mutations onto the internal nodes. For the region and host partitions, this means that we will infer the region or host onto internal nodes. Go ahead and select both boxes under "Ancestral State Reconstruction" for all 3 data partitions.



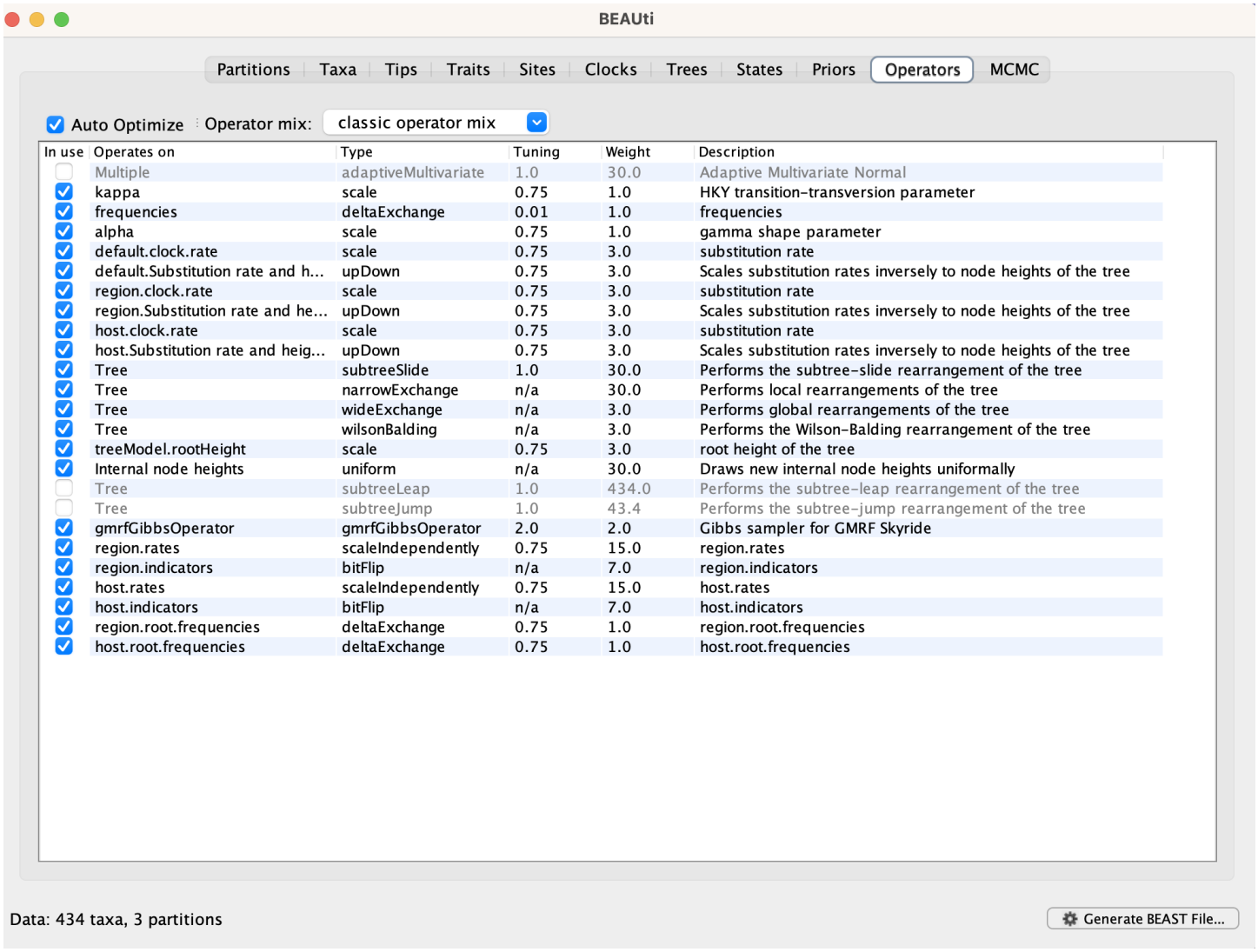
8. Picking priors is always at least a little bit tough. We want our priors to be reasonable, but we don't want them to constrain our analysis from exploring a variety of parameter values that could be probable. A good rule of thumb is that we want to pick broad, uninformative, but reasonable priors. Here are a few we can pick:

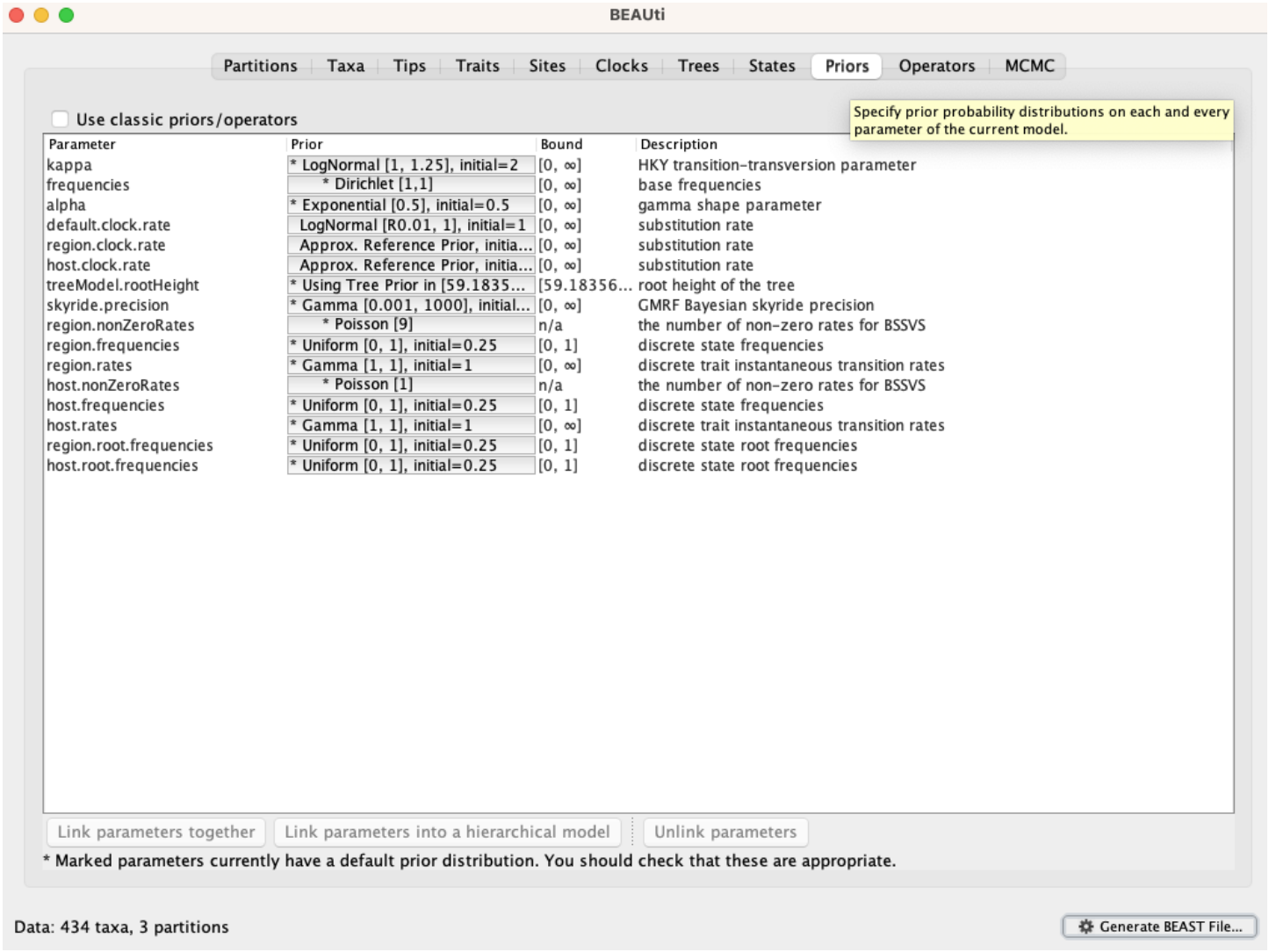
Clock rate: For the clock rate, we will use a lognormal distribution. Click on the "Mean/Stdev in real space" box, and check out the distribution as shown in the window below. We have a reasonable idea for what the clock rate should be for influenza, but we still don't want to influence the analysis too heavily. Leaving the defaults gives us a prior distribution with a 95% confidence interval of ~ 10^-7 to 10^-2. We should be almost positive that our real clock rate should fall within this range, so this is a reasonable range to set it.



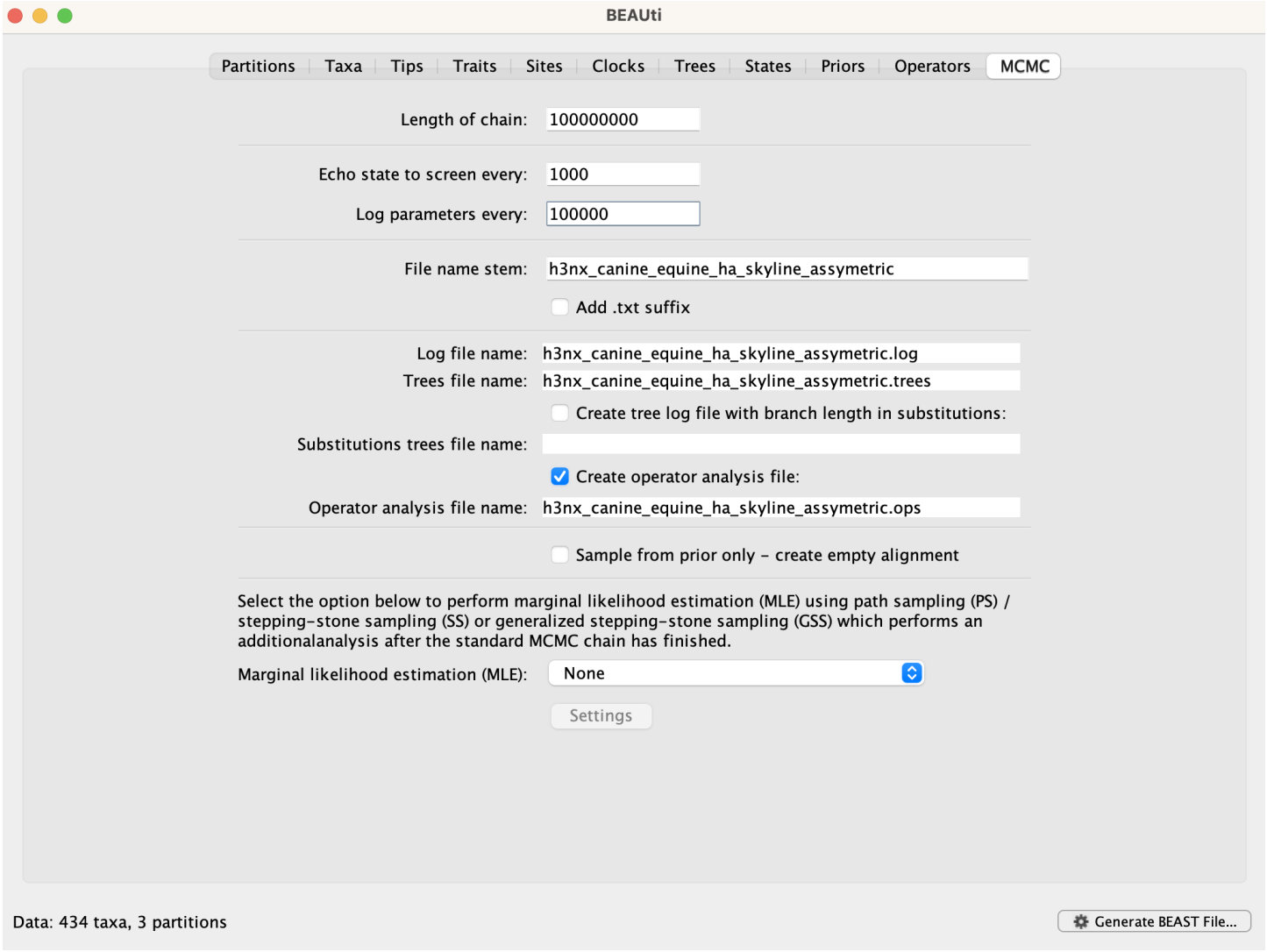
The rest: the rest of the priors we can leave as is. We don't have incredibly strong prior knowledge about these other parameters, and the defaults are reasonable. Leave as is.

9. Operators. The operators in BEAST specify how parameters are chosen/searched. The correct operators and their weights are usually determined by the scientists who developed and tested the models that are included in BEAST, so usually the defaults are pretty good. I have only ever changed an operator once, and it was when I was manually editing my xml to add in some analyses. You can read more about the specific types of operators and how they work in the BEAST documentation. The weight represents how frequently those different parameters will be updated/sampled. Higher weighted operators operate more frequently, while lower weighted ones operate less frequently. Usually parameters that are more complex/difficult to infer are assigned higher weighted operators. You'll see on your screen that parameters relating to the tree topology inference are weighted the highest, followed by inferring the rates for region and host. This is because the tree topology is the hardest thing to infer with the most parameters, so it requires the most operational time within the MCMC chain.

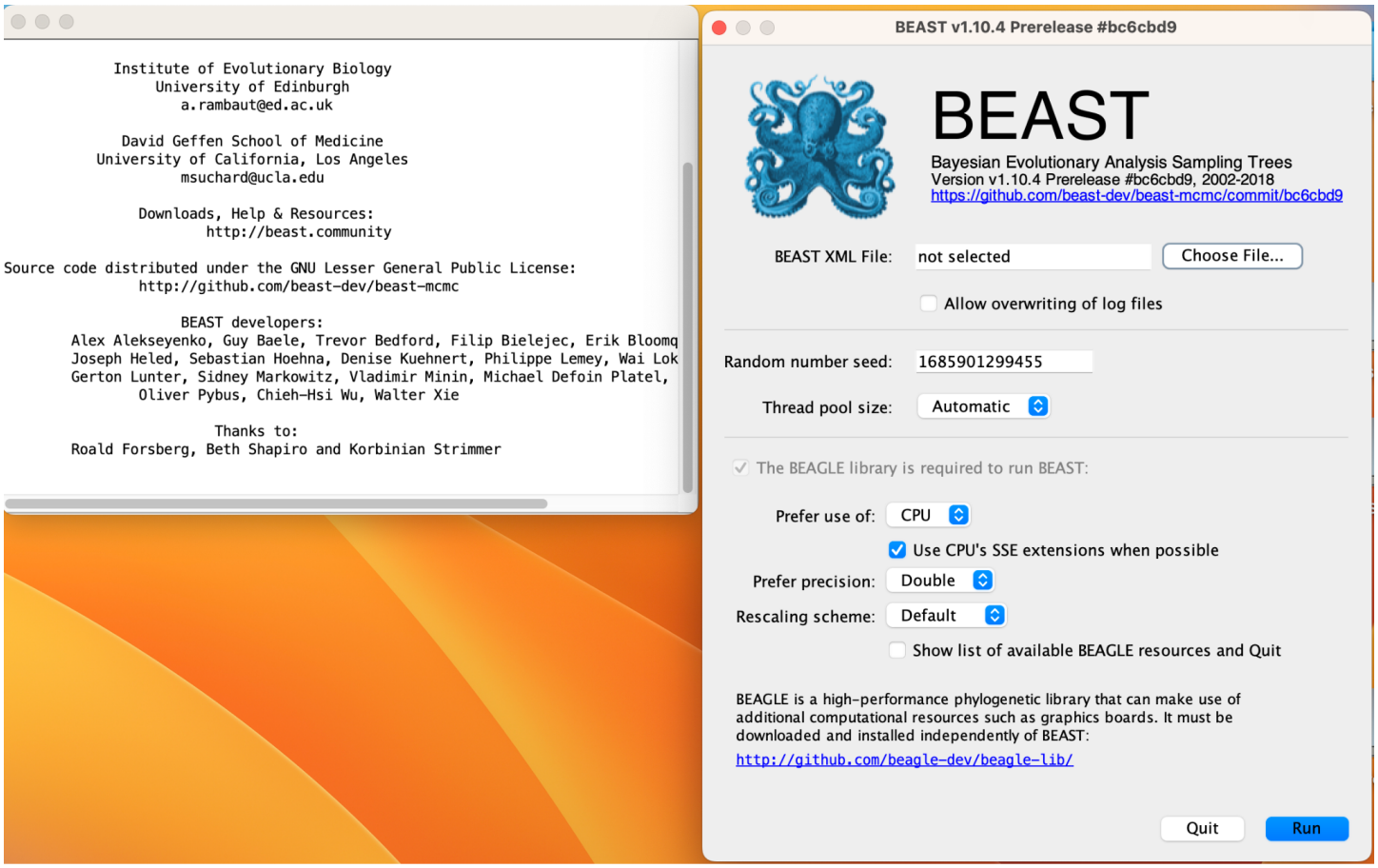

BEAUti
Partitions
Taxa
Tips
Traits
Sites
Clocks
Trees
States
Priors
Operators
MCMC
Auto Optimize
Operator mix:
classic operator mix
In use
Operates on
Type
Tuning
Weight
Description
Multiple
adaptiveMultivariate
1.0
30.0
Adaptive Multivariate Normal
kappa
scale
0.75
1.0
HKY transition-transversion parameter
frequencies
deltaExchange
0.01
1.0
frequencies
alpha
scale
0.75
1.0
gamma shape parameter
default.clock.rate
scale
0.75
3.0
substitution rate
default.Substitution rate and h...
upDown
0.75
3.0
Scales substitution rates inversely to node heights of the tree
region.clock.rate
scale
0.75
3.0
substitution rate
region.Substitution rate and he...
upDown
0.75
3.0
Scales substitution rates inversely to node heights of the tree
host.clock.rate
scale
0.75
3.0
substitution rate
host.Substitution rate and heig...
upDown
0.75
3.0
Scales substitution rates inversely to node heights of the tree
Tree
subtreeSlide
1.0
30.0
Performs the subtree-slide rearrangement of the tree
Tree
narrowExchange
n/a
30.0
Performs local rearrangements of the tree
Tree
wideExchange
n/a
3.0
Performs global rearrangements of the tree
Tree
wilsonBalding
n/a
3.0
Performs the Wilson-Balding rearrangement of the tree
treeModel.rootHeight
scale
0.75
3.0
root height of the tree
Internal node heights
uniform
n/a
30.0
Draws new internal node heights uniformally
Tree
subtreeLeap
1.0
434.0
Performs the subtree-leap rearrangement of the tree
Tree
subtreeJump
1.0
43.4
Performs the subtree-jump rearrangement of the tree
gmrfGibbsOperator
gmrfGibbsOperator
2.0
2.0
Gibbs sampler for GMRF Skyride
region.rates
scaleIndependently
0.75
15.0
region.rates
region.indicators
bitFlip
n/a
7.0
region.indicators
host.rates
scaleIndependently
0.75
15.0
host.rates
host.indicators
bitFlip
n/a
7.0
host.indicators
region.root.frequencies
deltaExchange
0.75
1.0
region.root.frequencies
host.root.frequencies
deltaExchange
0.75
1.0
host.root.frequencies
Data: 434 taxa, 3 partitions
Generate BEAST File...



10. MCMC chain. We're almost done! The last step is to specify how long you want to run the chain and how frequently you'd like to sample. A very common default is to run the chain for 50 million steps, sampling every 5,000. However, I've found that with this frequency of sampling, you can end up with enormous .trees output files and that these are very cumbersome to work with. BEAST analyses can be restarted in theory, but I always find it easier to just set a very long chain length, and then stop it prematurely if it converges faster than I anticipate. So for this analysis, we'll set the chain length to 100,000,000 generations, sampling every 100,000. As a rule of thumb, sampling ~0.1% of sampled steps will give you reasonable output files while sampling frequency enough to have a good sample of the posterior. Another good rule of thumb is that at the end, sampling ~1000 trees is a good target. Again, this keeps file sizes to a reasonable size while ensuring adequate sampling of the posterior. You can also specify output file names here. When you're done, click on "Generate BEAST file", and save it to a location on your laptop. The saved file will have a ".xml" extension, which is openable in a standard text editor. Feel free to open it up and take a look in Text Wrangler.



11. Start the analysis. Double click on BEAST v1.10.4 app to open it. This will open up a command line screen and a graphical user interface. To upload your xml file, click on “Choose File” and upload it. You’ll see a few options here, none of which we need to edit. The random seed number is simply a random set of numbers assigned to this particular job. You can also specify usage of CPUs and GPUs. In some older versions of BEAST, you had to select using BEAGLE, but now it is automatic. BEAGLE handles memory scaling in BEAST, and is necessary/very useful to use in basically all cases. Click on “Run” and it should begin!



Once the run starts, the screen should look like this:

```
       BEAST v1.10.4 Prerelease #bc6cbd9, 2002-2018
      Bayesian Evolutionary Analysis Sampling Trees
                  Designed and developed by
   Alexei J. Drummond, Andrew Rambaut and Marc A. Suchard

              Department of Computer Science
                  University of Auckland
                  alexei@cs.auckland.ac.nz

             Institute of Evolutionary Biology
                  University of Edinburgh
                     a.rambaut@ed.ac.uk

              David Geffen School of Medicine
            University of California, Los Angeles
                     msuchard@ucla.edu

                Downloads, Help & Resources:
                        http://beast.community

Source code distributed under the GNU Lesser General Public License:
                http://github.com/beast-dev/beast-mcmc

                      BEAST developers:
       Alex Alekseyenko, Guy Baele, Trevor Bedford, Filip Bielejec, Erik Bloomquist, Matthew Hall,
       Joseph Heled, Sebastian Hoehna, Denise Kuehnert, Philippe Lemey, Wai Lok Sibon Li,
       Gerton Lunter, Sidney Markowitz, Vladimir Minin, Michael Defoin Platel,
               Oliver Pybus, Chieh-Hsi Wu, Walter Xie

                       Thanks to:
       Roald Forsberg, Beth Shapiro and Korbinian Strimmer

Using BEAGLE library v4.0.0 (PRE-RELEASE) for accelerated, parallel likelihood evaluation
2009-, BEAGLE Working Group - https://beagle-dev.github.io/
Citation: Ayres et al (2019) Systematic Biology 68: 1052-1061 | doi:10.1093/sysbio/syz020


Random number seed: 1685901529222


Loading additional development parsers from development_parsers.properties, which is additional set of parsers only available for development ver
sion ...
Parsing XML file: h3nx_canine_equine_ha_skyride_assymetric-fixed-loggers.xml
  File encoding: UTF8
Looking for plugins in /Users/lhmoncla/plugins

Read alignment: alignment
  Sequences = 434
      Sites = 1740
   Datatype = nucleotide
Site patterns 'patterns' created from positions 1-1740 of alignment 'alignment'
  pattern count = 1740
Read attribute patterns, 'region.pattern' for attribute, region
Read attribute patterns, 'host.pattern' for attribute, host
```

And eventually, it will start running and look like this. Notice on the far right hand column, you will see a printout of how quickly the analysis will run. This one on my laptop will take an expected 0.23 hours per million generations, or ~1 day to get to 100 million generations.

```
 1.10. Virus Evolution. vey016. DOI:10.1093/ve/vey016
Using BEAGLE likelihood calculation library:
        Ayres et al  (2012) BEAGLE: a common application programming inferface and high-performance computing library for statistical phylogeneti
cs. Syst Biol. 61, 170-173. DOI:10.1093/sysbio/syr100

TREE DENSITY MODELS
Skyride coalescent:
        Minin VN, Bloomquist EW, Suchard MA (2008) Smooth skyride through a rough skyline: Bayesian coalescent-based inference of population dyna
mics. Mol Biol Evol. 25, 1459-1471. DOI:10.1093/molbev/msn090

SUBSTITUTION MODELS
HKY nucleotide substitution model:
        Hasegawa M, Kishino H, Yano T (1985) Dating the human-ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol.. 22, 160-174
Discrete gamma-distributed rate heterogeneity model:
        Yang Z (1994) Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. J. Mol.
Evol.. 39, 306-314
Complex-diagonalizable, irreversible substitution model:
        Edwards CJ, Suchard MA, Lemey P, Welch JJ, Barnes I, Fulton TL, Barnett R, O'Connell TC, Coxon P, Monaghan N, Valdiosera CE, Lorenzen ED,
 Willerslev E, Baryshnikov GF, Rambaut A, Thomas MG, Bradley DG, Shapiro B (2011) Ancient hybridization and an Irish origin for the modern polar
bear matriline. Current Biology. 21, 1251-1258

PRIOR MODELS
CTMC Scale Reference Prior model:
        Ferreira MAR, Suchard MA (2008) Bayesian analysis of elapsed times in continuous-time Markov chains. Canadian Journal of Statistics. 36,
355-368


# BEAST v1.10.4 Prerelease #bc6cbd9
# Generated Sun Jun 04 13:58:49 EDT 2023 [seed=1685901529222]
# /Users/lhmoncla/src/csu-infectious-disease-wildlife-course/beast-lab/h3nx-canine-equine/h3nx_canine_equine_ha_skyride_assymetric-fixed-loggers.
xml
# keywords: skyride
state   Joint           Prior           Likelihood      age(root)       default.clock.rate      region.clock.rate       host.clock.rate region.no
nZeroRates    host.nonZeroRates
0       -184277.9366    -24267.0602     -160010.8764    1444.87         1.00000         1.00000         1.00000         90      2       -
Underflow calculating likelihood. Attempting a rescaling...
1000    -78782.1905     -5323.1122      -73459.0784     1712.06         0.25470         0.23333         0.52394         79      2       -
2000    -67754.7160     -3969.1910      -63785.5250     1858.83         0.12244         0.11946         0.29555         55      2       -
3000    -64146.2391     -4048.6760      -60097.5632     1895.51         2.85093E-2      4.50713E-2      0.18733         43      2       -
4000    -61153.5705     -3633.1219      -57520.4486     1901.12         9.84405E-3      3.39147E-2      6.33413E-2      38      2       -
5000    -59840.8966     -3757.1825      -56083.7141     1904.52         8.78955E-3      3.36517E-2      1.34273E-2      33      2       -
6000    -58468.9208     -3640.7916      -54828.1292     1905.37         8.66187E-3      2.83237E-2      1.00199E-2      29      2       -
7000    -56882.5124     -3694.7916      -53187.7208     1905.88         7.85257E-3      2.26064E-2      1.16233E-2      28      2       -
8000    -56077.6372     -3691.0552      -52386.5821     1907.44         7.51445E-3      2.51472E-2      9.26071E-3      26      2       -
9000    -55185.2396     -3814.0480      -51371.1916     1909.97         4.54493E-3      2.66625E-2      1.09681E-2      27      2       -
10000   -54025.1232     -3993.5226      -50031.6006     1918.84         3.90258E-3      2.55705E-2      9.82184E-3      28      2       -
11000   -52783.7338     -3697.4834      -49086.2504     1923.54         3.95794E-3      2.02276E-2      6.29722E-3      28      2       0.23 hour
s/million states
12000   -52420.9951     -4032.6853      -48388.3098     1925.05         3.89878E-3      2.29289E-2      9.01973E-3      28      2       0.24 hour
s/million states
13000   -51663.5215     -3789.3068      -47874.2147     1925.91         3.84002E-3      1.99759E-2      7.93876E-3      30      2       0.23 hour
s/million states
14000   -51134.3529     -3775.7640      -47358.5890     1925.54         3.69647E-3      2.20645E-2      8.67158E-3      28      2       0.23 hour
s/million states
15000   -50590.1870     -3933.3031      -46656.8839     1926.39         3.52161E-3      2.128E-2        5.24574E-3      27      2       0.24 hour
s/million states
```

Note: if the above didn't work for you for some reason, you may have some sort of java version issue. To get around this, you can always open up a Terminal window, and drag the beast executable within the "BEAST v1.10.4/bin/beast" into the terminal window, followed by the path to your xml. That will also launch BEAST successfully.

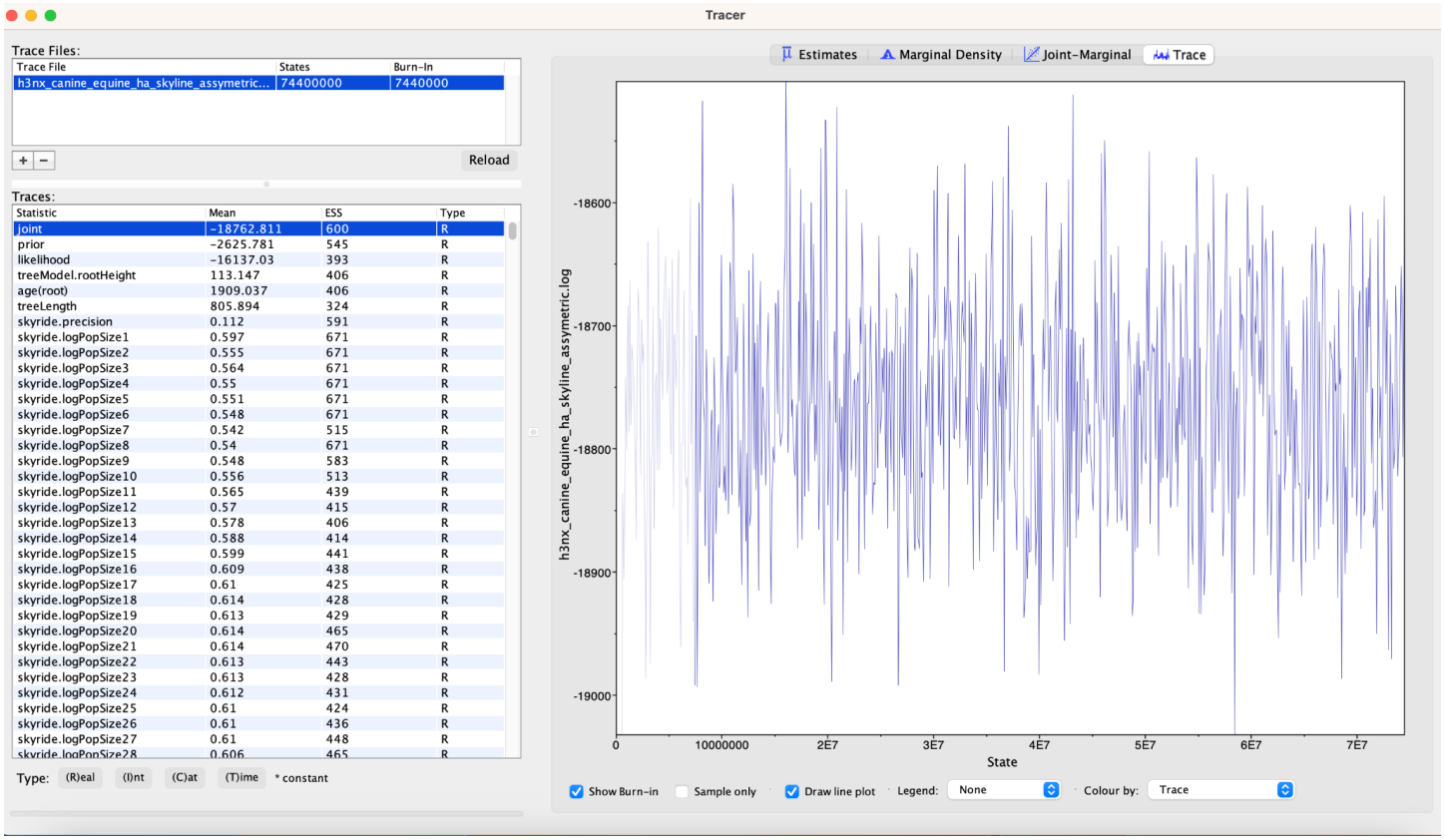
12. Finally, although we won't do this here, it is always a good idea to run your BEAST runs in replicates, and then combine and compare the results. The reason this is necessary is that MCMC chains are random, so different chains initiating from the same xml file may start and traverse different parameter spaces. If multiple chains converge on very similar parameter distributions, this is a good indication that those distributions are frequently inferred as the most probable. If the different chains converge on very different results, that tells you that there are multiple combinations of parameters that are compatible with the alignment and model. Having an accurate sense of uncertainty is scientifically important, and a real benefit of Bayesian analyses. I usually launch 3 independent replicates of the same xml, simply by copying those xmls and renaming them something like "it1-xmlname.xml", "it2-xmlname.xml", etc… Most computational clusters have batch systems that make this easy.

Part 2: Interpreting the analysis

1. We don't have a whole day to wait for this analysis to run, so I've run this beforehand on our cluster and uploaded the results for us to look at. Go to the folder: "Beast-lab/output-files/Analysis-1-DTA-skyride/". You'll notice 2 folders here: 2M-steps-checkpoint and 74M-steps-checkpoint. I saved results from this analysis as it sampled along the chain so that you could see what chains look like early and late. The 2M folder includes results stopped after 2 million generations, which is almost always going to be too short for the run to have converged. The 74M folder contains results after 74 million generations, which is fairly long. Go ahead and unzip both folders, and navigate into one of then. You'll see that I ran 3 independent iterations of the aIn each iteration folder, there would be the following output files:

1. "h3nx_canine_equine_ha_skyline_assymetric.log": this is the primary log file that reports the results of the analysis. Along the MCMC chain, parameters are sampled and more probable values are retained in the chain. We specified a very long chain (100 million generations), sampling every 100,000. So the log file contains every single parameter sampled every 100,000 generations. You can open this as a text file and see that it is just a massive, tab-delimited file, with a ton of columns. Each row represents a recorded step.
2. "H3nx_canine_equine_ha_skyline_assymetric.trees": this is our output posterior tree distribution. This file will be large and contains hundreds of sampled trees. You can open this file in a tree viewer or with a text editor.
3. "H3nx_canine_equine_ha_skyride_assymetric-fixed-loggers.xml": this is our input xml
4. A series of other log files. These are are all subsets of the main log file that contain only a subset of those parameters. We can ignore these log files for this activity.

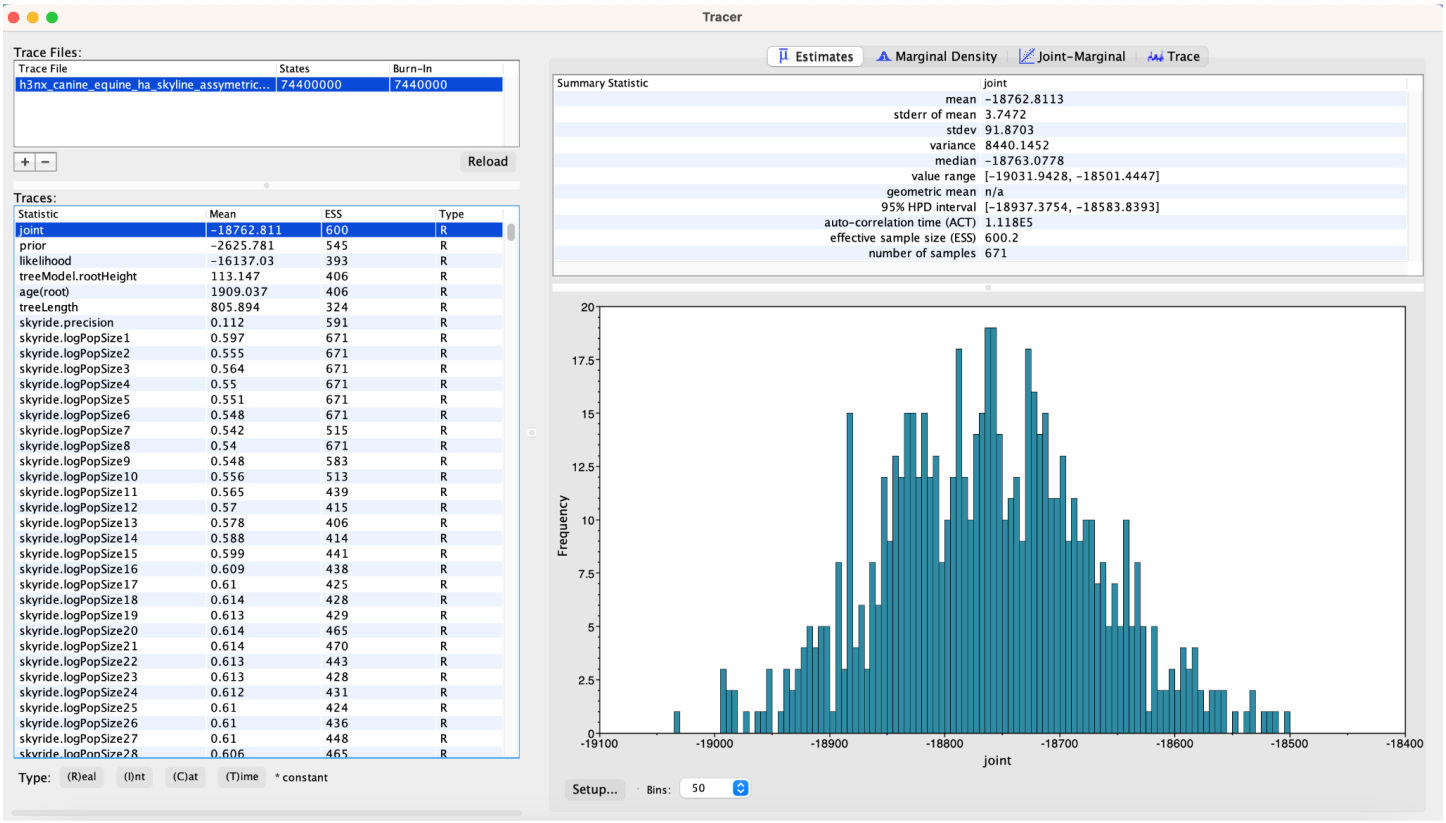
2. To look at the results of our run, first open up "Tracer". Tracer is a piece of software developed to visualizing BEAST MCMC chains. Navigate to the "74M" generations folder and into "it1". Drag and drop the h3nx_canine_equine_ha_skyline_assymetric.log" log file onto the Tracer main screen. It should look something like this:



There is a lot going on here, so we will unpack it. The top, righthand box gives an overview of the analysis. It shows the file you've loaded, and how many steps in the chain the analysis has gotten to. In this screenshot, the number of steps, or states, is ~74 million. Right below that are the logged values for each parameter we've sampled. Each parameter is listed under the "Statistic" column, and the mean across the estimates is shown in the "Mean" column. For each column in the log file, the mean represents the mean value across that column.

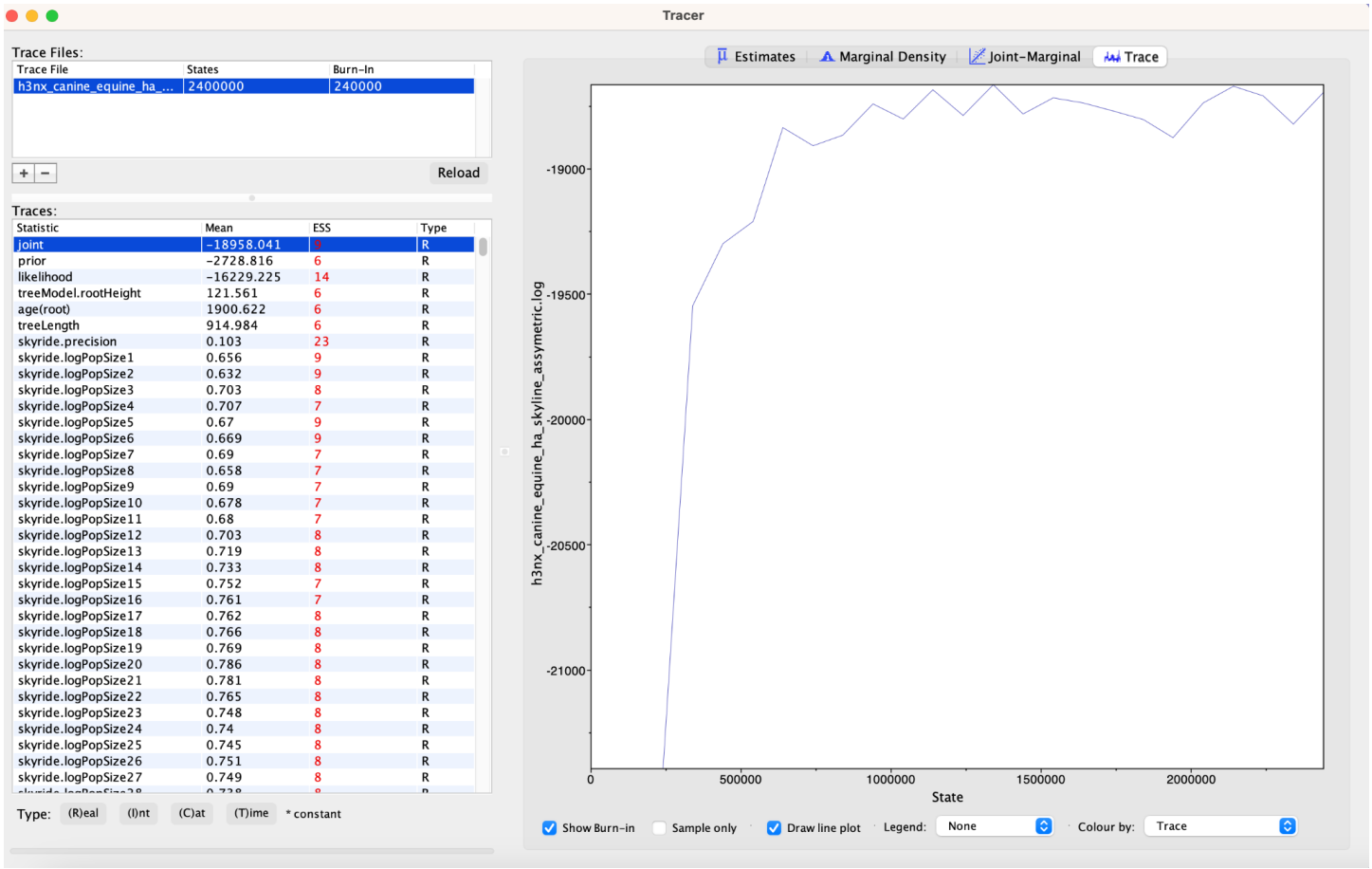
The next column is the ESS value. ESS stands for effective sample size and represents how well "mixed" the chain was. Recall that one important feature of a bayesian analysis is that the chains mix appropriately, and explore a wide range of parameter space. ESS values capture how well that has been achieved. The actual "traces" are shown in the panel to the right. A good, well-mixed sample looks like the screenshot above: it looks like a series of wide lines. In general, for any parameter you really want to estimate accurately, you want to achieve an ESS of at least 200. Often, you can increase ESS values by simply running the chain for longer. However, if the ESS is persistently low, that can indicate that your mixing is off, and that either your model may be a bad fit or that you need to alter the operators.

You can toggle the options in the top panel to see histograms and density plots of the inferred distributions rather than the traces. If instead we wanted to look at the histogram, it would look like this:

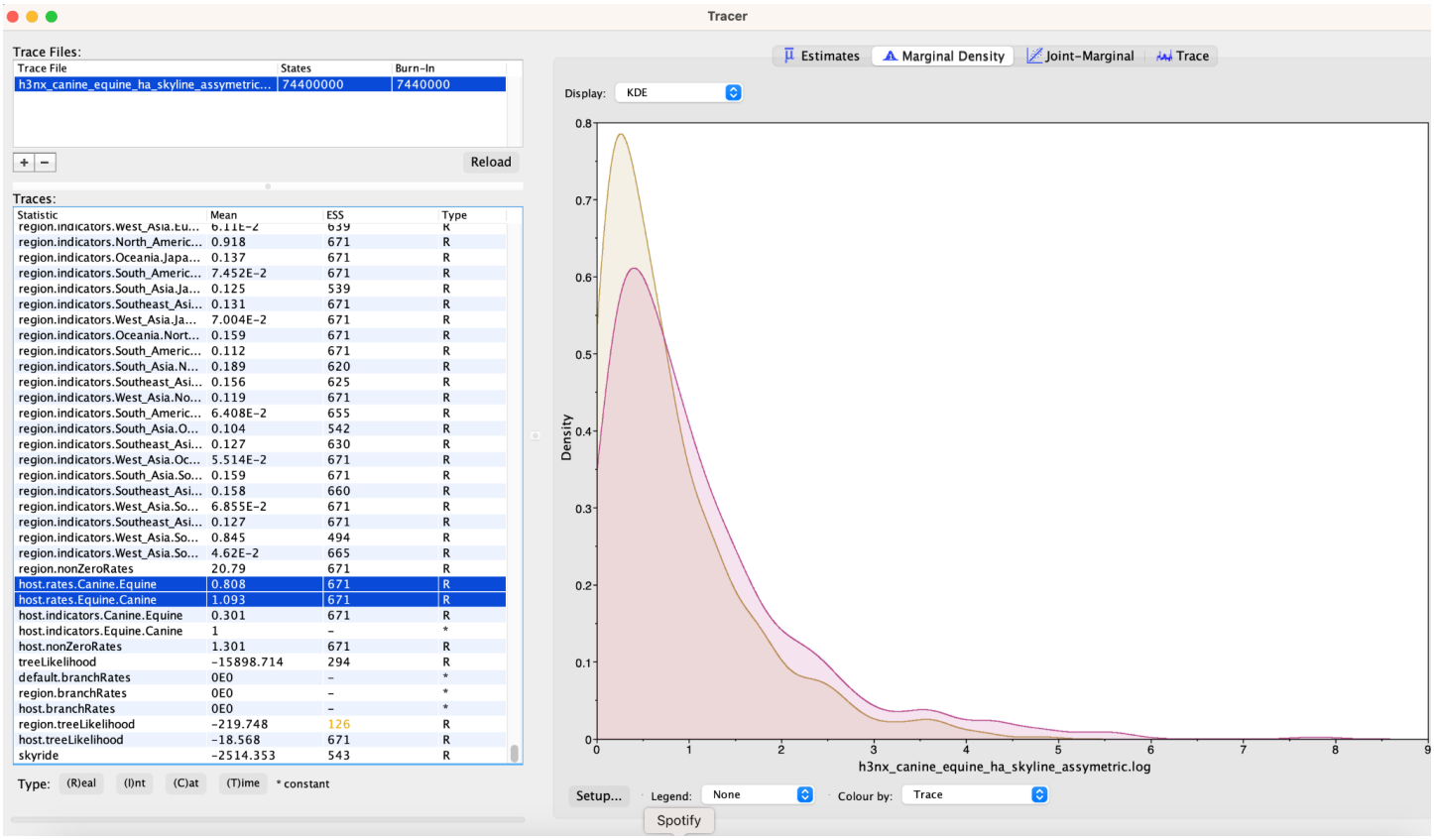


Finally, the top panel has “Burnin” specified. BEAST automatically defaults to a 10% burning, but you can and should change this to achieve better mixing/ESS values. You’ll sometimes need to discard less than 10% as burnin, and sometimes you’ll need to discard more. I’ve discarded burnins as high as 50% before, in cases where the model was complex and it took a very long time to find a good area of search space. In this instance, our run has converged really nicely, so leaving as 10% burnin should be totally fine.

5. As a comparison, drag in one of the log files from the 2M generations checkpoint folder. Notice that all of these ESS values are in red, and are very low. This run has absolutely not converged yet. However, by letting it run for another 70M or so generations, the run has converged nicely. You’ll also notice that a lot of these trace files show an initial phase where the values are all really low, and then a sudden point where the algorithm has found a better search space and the traces quickly ascend up to a higher value. This is a very common signal, and tells you that your analysis is on its way to convergence.

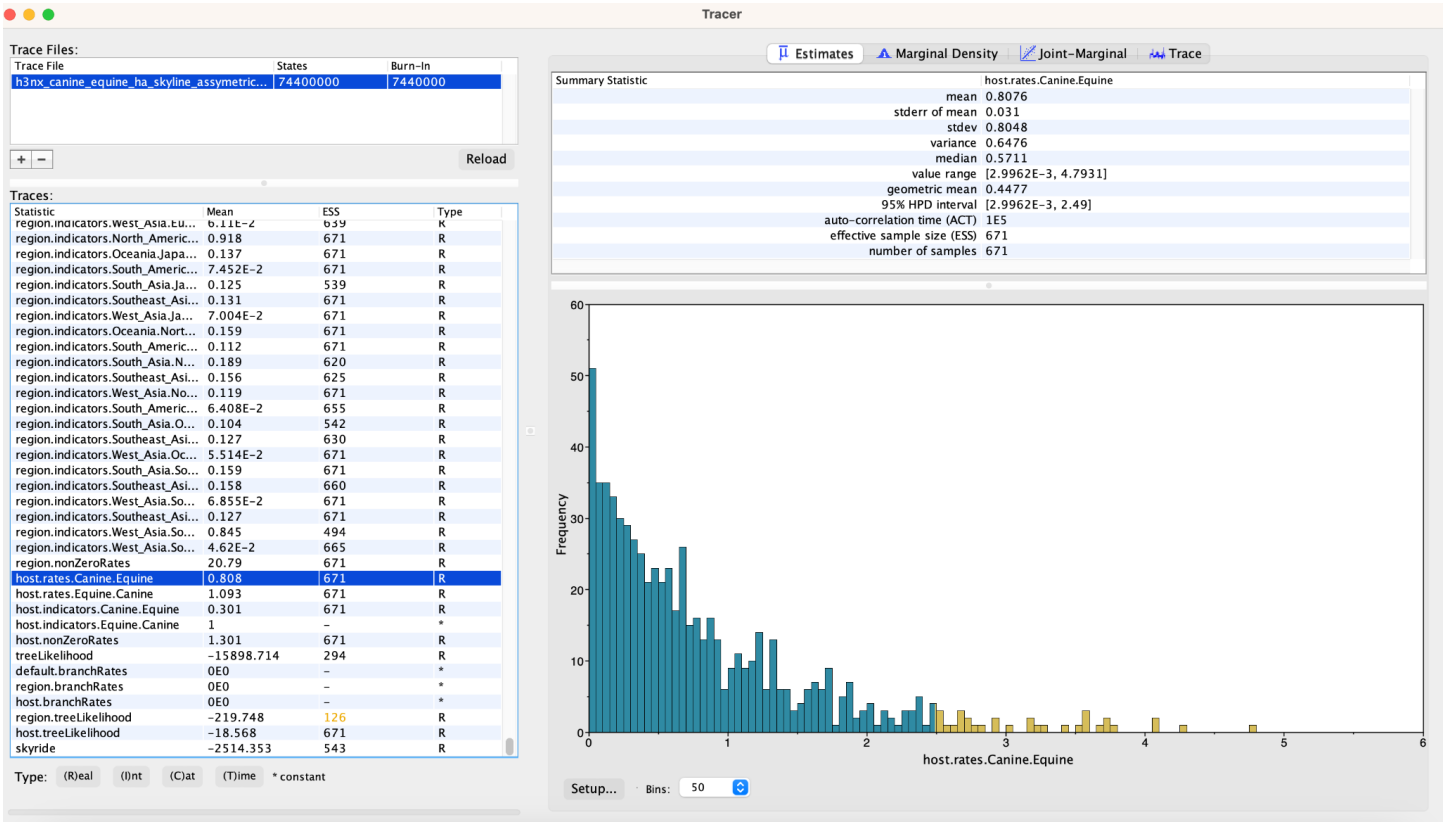


6. Now, let's take a look at our estimated parameters. Scroll to the very bottom of the screen, and click on "host.rates.Canine.Equine" and "host.rates.Equine.Canine". These indicate the inferred rates of transmission from dogs to horses and vice versa. Select both, then pick "marginal" for the display. This shows us our distributions.

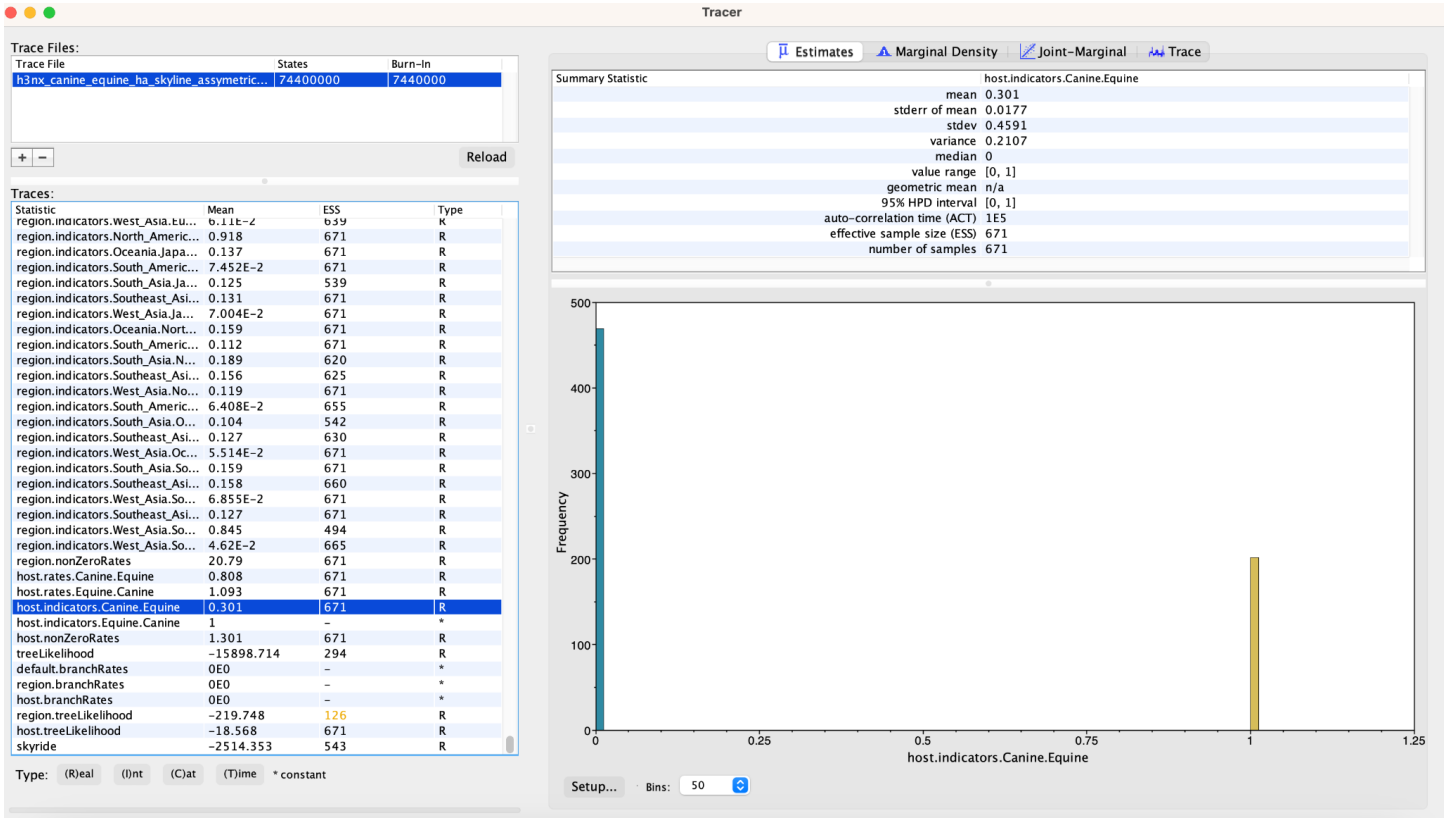


You can see that we've estimated very similar rates for these hosts. This makes sense for this analysis, because we've only included the portion of the tree that is the equine lineage that was transmitted to dogs. These rates tend to be reported in transitions per lineage per year. So this means that on average, on the path from the root to a tip, there are an average of 1 transition events from horses to dogs.

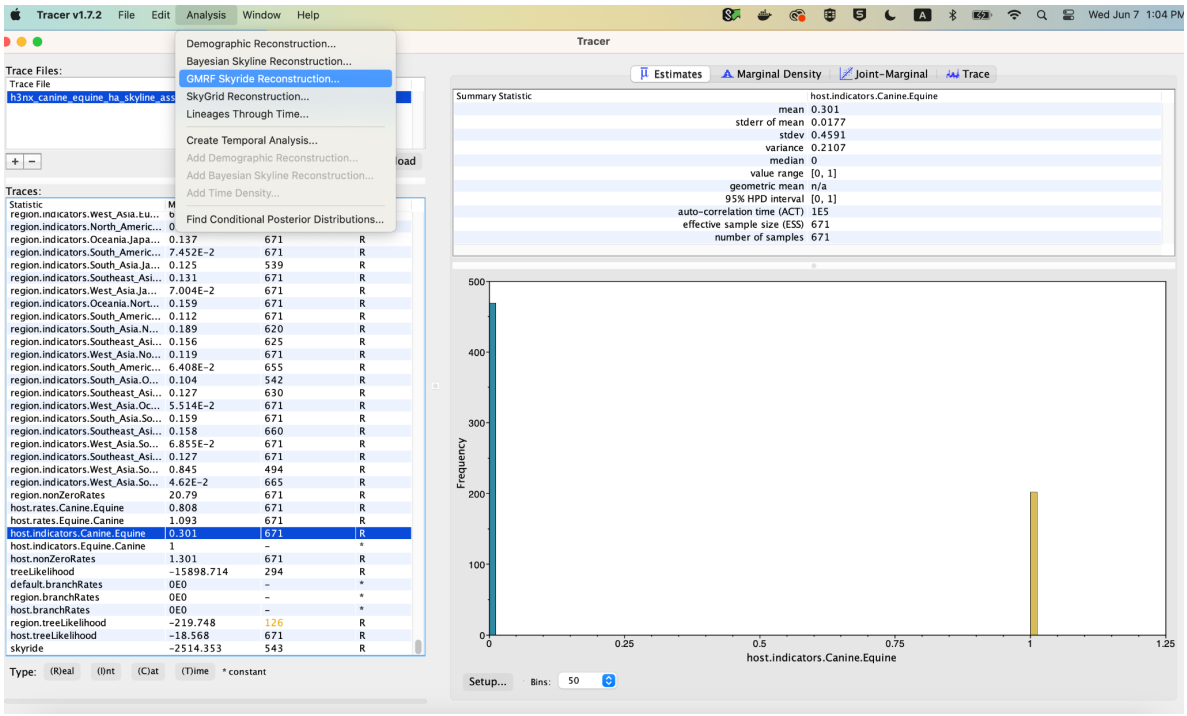
7. Now lets look at those rates in a different way. Click on just "host.rates.Canine.Equine" and click on "u estimates" for the view. That will show a histogram. Notice how the histogram is bumped up against 0. It looks as if the estimate almost "wants" to be 0, or negative.



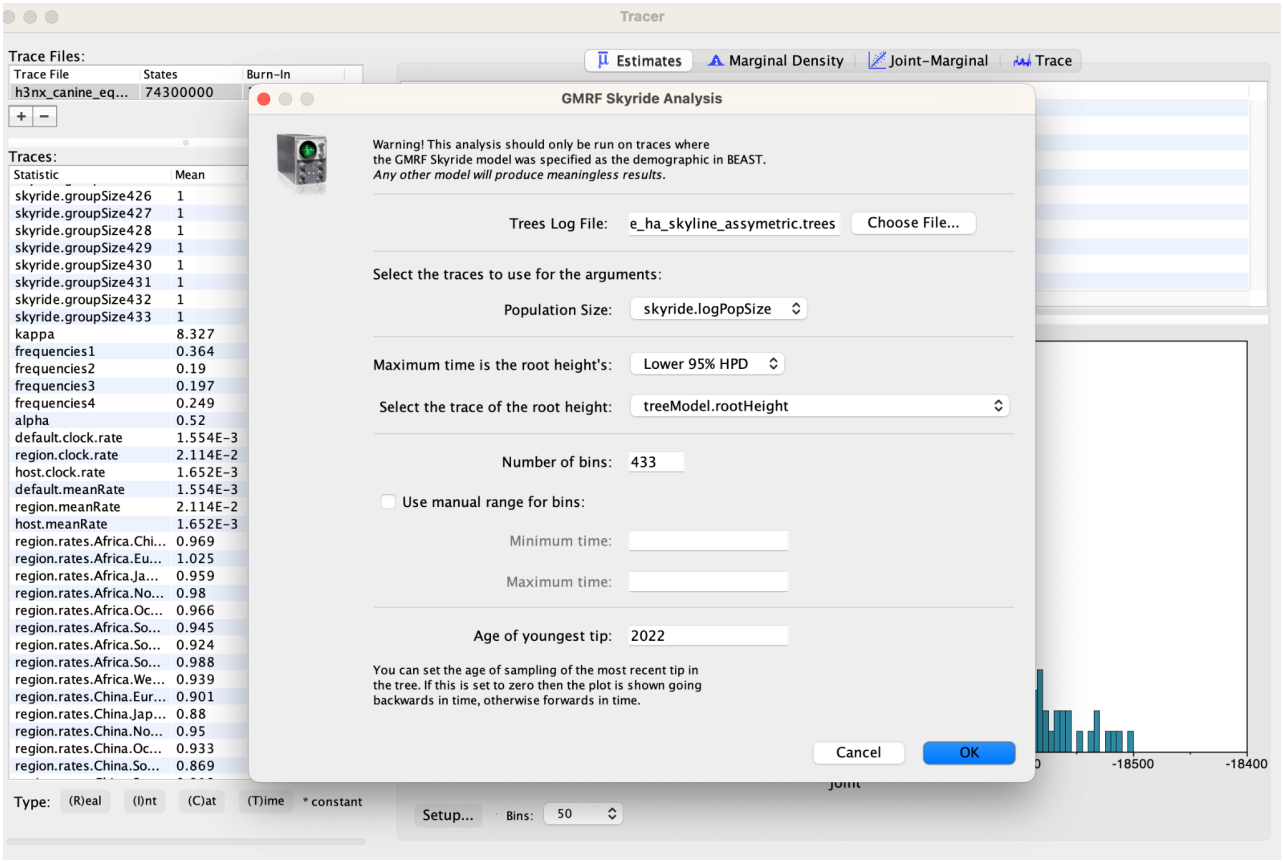
A negative transmission rate doesn't make sense, but a 0 transmission rate does. Because we ran a BSSVS analysis, we also included indicator variables for each of these host rates. These indicators are a measure of how certain we are that those host rates should have a value and be included in the model. Those values are reported in the "host.indicators.Canine.Equine" and "host.indicators.Equine.Canine". Click on the "host.indicators.Canine.Equine" first. Notice that for this parameter, it is an indicator variable, so the only possible values we sample are 0 and 1 (indicating on and off). Notice that although our mean value for the canine to equine indicator is 0.3, the 0 value is actually more probable/has greater support. This tells us that the most parsimonious model for this data only includes a rate from equine to canine, and not from canine to equine. This makes a lot of sense given the tree topology that we observed in the Nextstrain lab. In the equine lineage, we see a transmission event to dogs, but don't directly see transmission from dogs back to horses. In contrast, if you look at the equine to canine indicator, it is very strongly inferred as 1. The MCMC chain essentially never samples a 0, so we can be pretty confident that this is a real, meaningful rate.



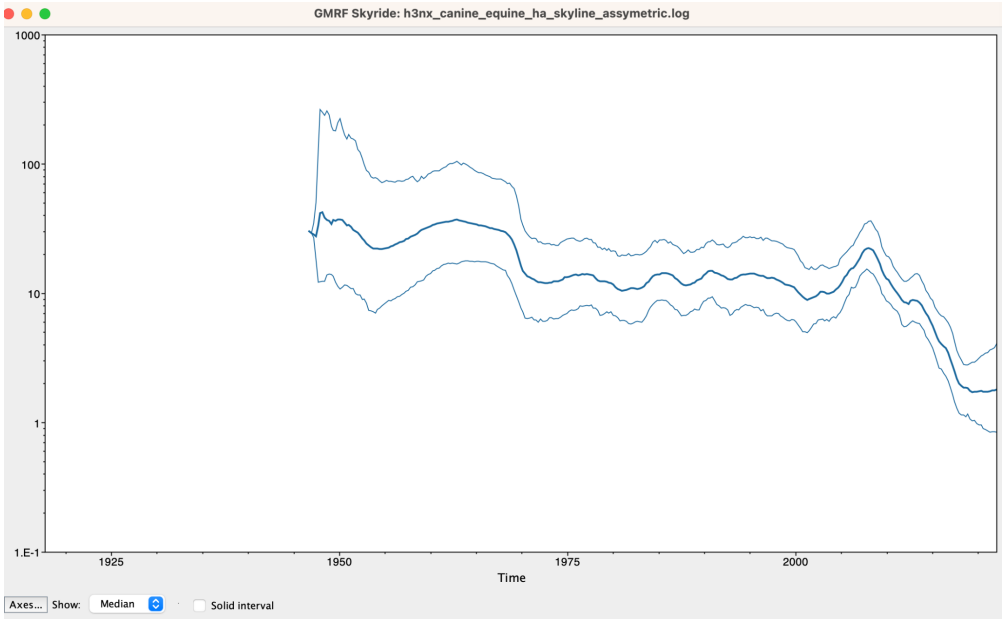
8. Another parameter we wanted to estimate was the rate of population growth in this lineage. In our xml, we specified a skyride population model. We can now summarize this population growth by running a skyride analysis in Tracer. To do so, go to the top of the screen, and select “Analysis” -> “GMRF Skyride Reconstruction” .



Select your .trees output file from our outputs. For the age of youngest tip, put “2022”, since that is our most recent tip in the tree. For the number of bins, annotate 433. Our analysis was run with 433 bins, so we can use that number. These arguments don’t really impact the analysis, but they do convert the timescales on the resulting figure into actual, intepretable time, which is easier to read. Click ok. It should start running the analysis. If you get an error about the number of logged states not matching, let me know! This will take a few minutes to run.



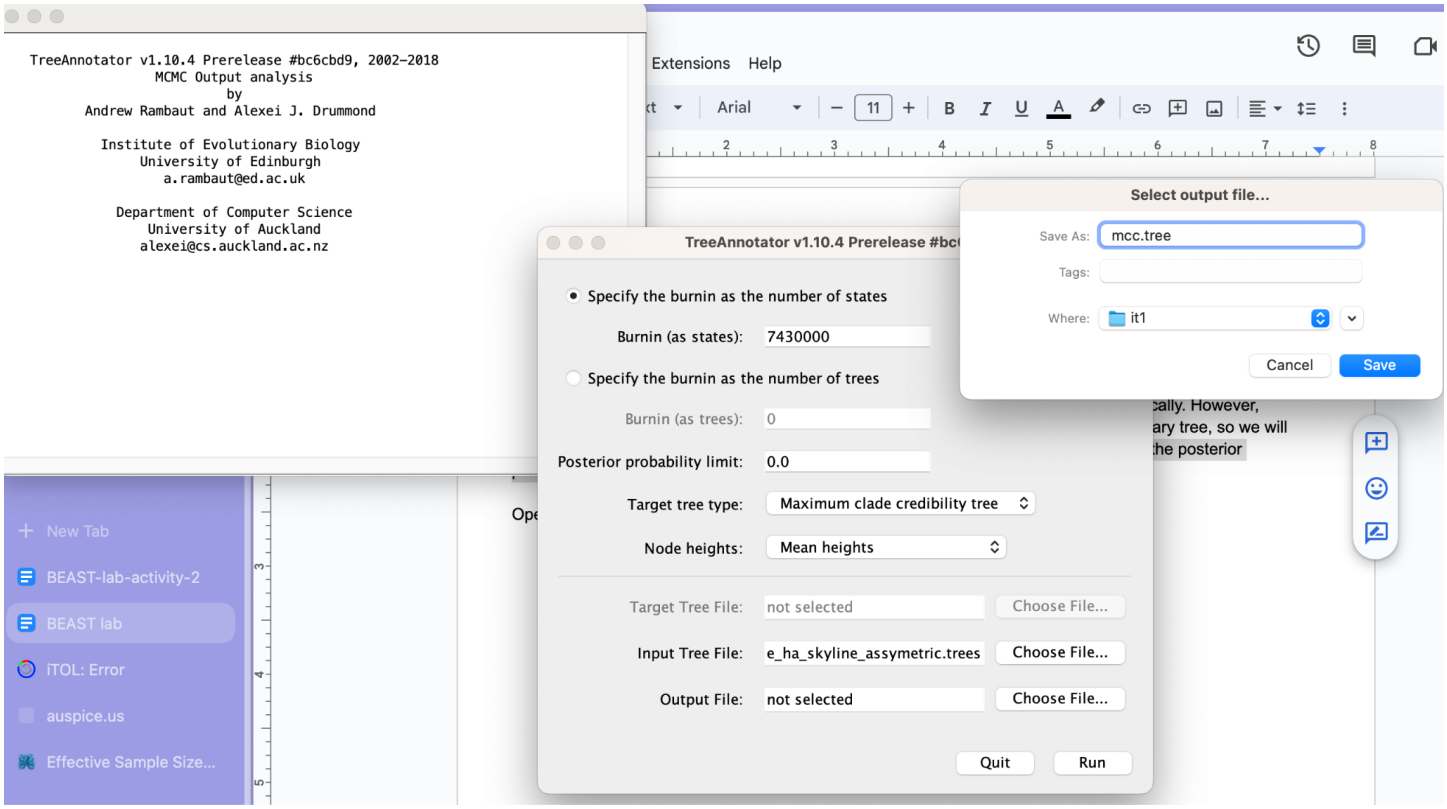
Once it finishes running, the output will look like this:



This analysis is telling us that the population size underwent a small expansion in approximately 2010, then rapidly declined. After we explore our tree, we will determine whether that makes sense!

9. Let's generate a summary tree. Our .trees output file contains our entire posterior set of trees. This is a really great feature of BEAST because you can iterate over this posterior and make up new statistics to estimate parameters you may want to estimate but didn't get logged in BEAST automatically. However, posterior sets of trees are a nightmare to visualize. Usually, we want some sort of summary tree, so we will generate a maximum clade credibility tree (mcc tree), which is a summary tree in which the posterior probabilities of each clade are maximized.

Open up Tree Annotator. Select your ".trees" file. To specify your burnin, BEAST 1 requires that you specify this as the number of states. This is exactly equivalent to the number that appears in your burnin box in Tracer. So here, we'll keep it at 10%, which is ~7 million states (7430000). Select "Maximum clade credibility tree", and "mean heights" for node heights. Finally, select a name for your MCC tree, and click "save".

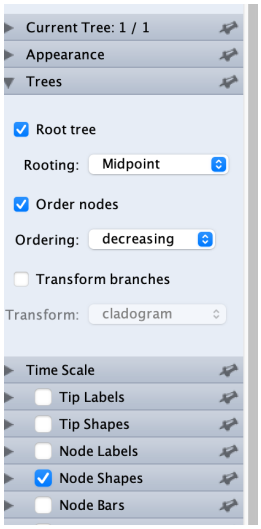


As the analysis is running, it will look like this: Notice that you will get a printout of how many trees are in the .trees file and will be analyzed, and how many are removed as burnin. Generally, getting anywhere from 500-1000 trees will give you a nice sample.

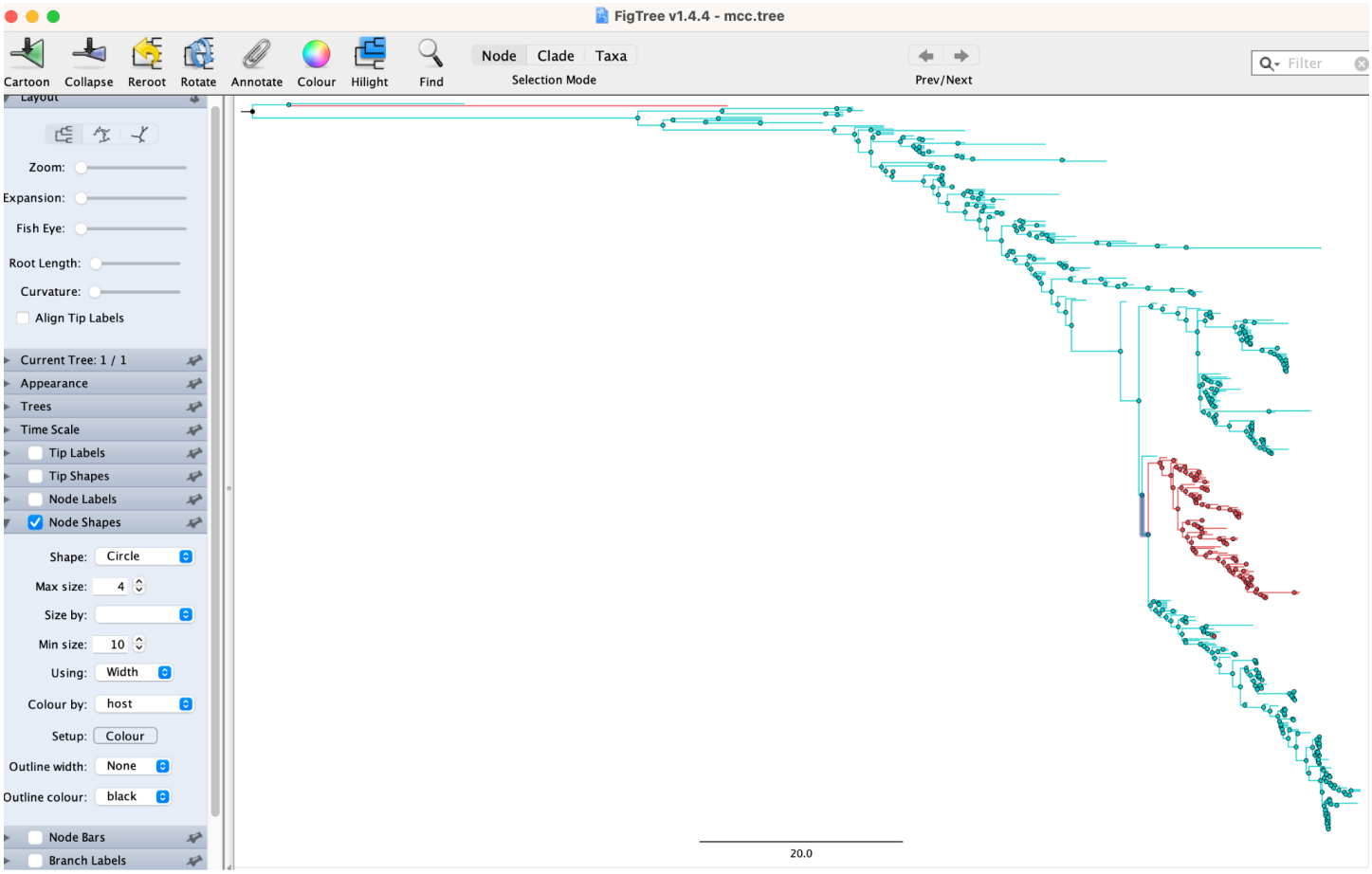
```
    TreeAnnotator v1.10.4 Prerelease #bc6cbd9, 2002-2018
                  MCMC Output analysis
                           by
          Andrew Rambaut and Alexei J. Drummond

            Institute of Evolutionary Biology
                 University of Edinburgh
                    a.rambaut@ed.ac.uk

              Department of Computer Science
                  University of Auckland
                 alexei@cs.auckland.ac.nz


Reading trees (bar assumes 10,000 trees)...
0              25             50             75             100
|--------------|--------------|--------------|--------------|
****

Total trees read: 744
Ignoring first 7430000 states (75 trees).
Total unique clades: 9941

Finding maximum credibility tree...
Analyzing 669 trees...
0              25             50             75             100
|--------------|--------------|--------------|--------------|
************************************************************

Best tree: STATE_57500000 (tree number 576)
Highest Log Clade Credibility: -403.1671659351885
Collecting node information...
0              25             50             75             100
|--------------|--------------|--------------|--------------|
************************************************************

Annotating target tree...
Writing annotated tree....
Finished - Quit program to exit.
```

10. Finally, let's look at our summary tree. We will need to download FigTree, which is a piece of software for visualizing trees. Download it from here: https://github.com/rambaut/figtree/releases and install.

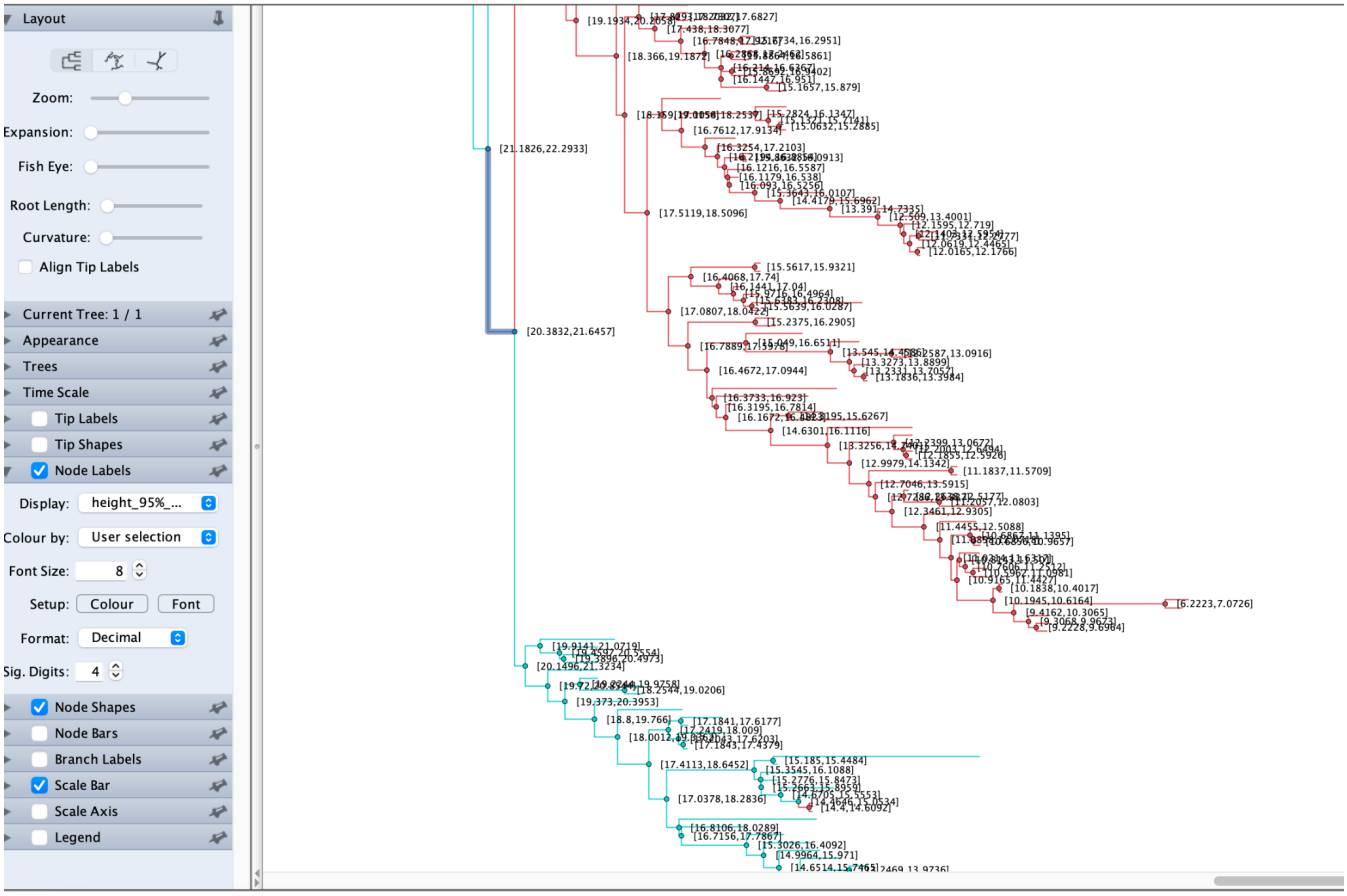
11. Now, open your mcc tree in Figtree. The default settings in Figtree are horrible and really hard to read. To make it easier to visualize, go to the lefthand panel and select a few options:
    a. First, under "Trees", click on "root tree" and "Midpoint". Then, click on "order nodes", order "decreasing". Finally, uncheck Tip labels.



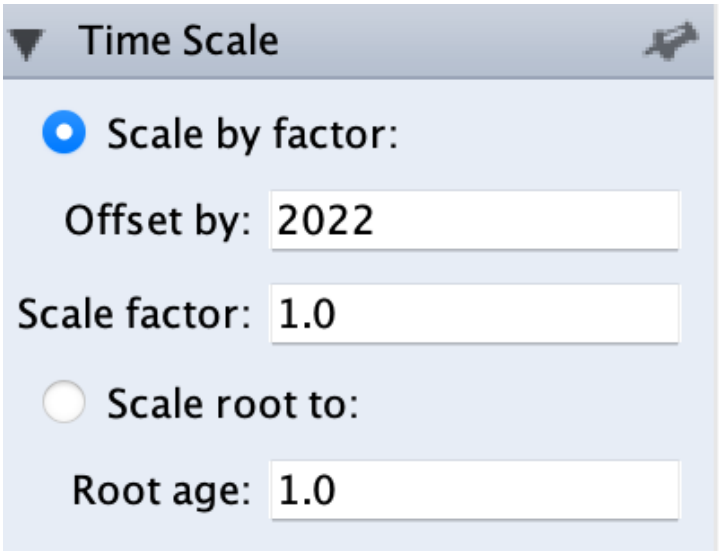
12. Now, let's color this tree by host group. Under "Appearance", click on "colorby" and "host". Finally, under "Node Shapes", click on "circle" and colorby host. You should now see a tree like this:



13. Notice that we have inferred the host trait onto internal nodes on this phylogeny. This tree is showing us that there was a single transmission event from horses into dogs. To determine when this occurred, zoom into the dog introduction, and select "Node labels" -> "height". This shows the inferred age of that node being ~21 years old. If we instead choose "height 95% HPD", this will show us the 95% highest posterior density estimate for the node age. We see that the model is estimating that this introduction occurred sometime between 20-22 years ago.



14. From looking at this tree, do our population size reconstructions make sense? Why do we think that the results are the way they are? To help with this, click on “Time Scale” and select “Scale by factor” and set to 2022. This will convert your timeline into real time.



15. Play around with the region geographic inference. How rapidly and frequency are H3N8 viruses moving between geographic regions?

16. To finish this out, think about everything we’ve estimated and what we can conclude from this analysis. What do all of these factors tell us about H3N8 transmission between horses and dogs? How frequent

is transmission, and is transmission bi-directional? When did these events occur? How are these viruses transmitting across geographic areas? Finally, did all 3 of our MCMC chains estimate similar parameter values?

An alternative visualization option: IcyTree

Navigate to https://icytree.org/. IcyTree is a cool piece of software that works really well with BEAST trees. It is my go-to software for flipping through posterior tree sets, although it only works for that purpose for BEAST 2 trees. However, it will visualize our mcc tree perfectly well, although it can take a long time to load tree files that are large. Drag and drop your mcc onto the web page, and then change around the view options with the "Style" pane.