**Computer Lab 8 – Coalescent Theory and Extended Bayesian Skyline Methods**

**Conservation Genetics (BIOL 4174 / 5174)**

Part I – Simulating the Coalescent in R

We will first explore the effects of population demography using some simulation tools implemented in R. This activity should take you 10-15 minutes at the most.

Open R in your preferred environment (e.g. R Studio), and first install the packages we will be using:

install.packages(c(“coalescentMCMC”, “adephylo”, “scrm”, “coala”))

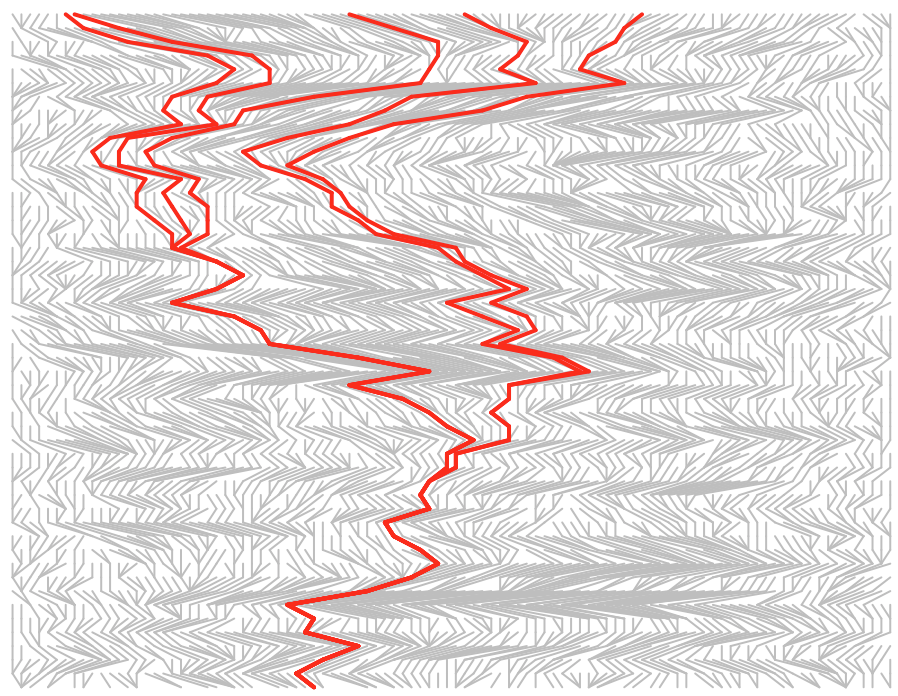
Next, you will need to load these packages (using the ‘library()’ command), and load some functions that I have provided for you. First, make sure your working directory is set to the location of your Lab8 files (e.g. ~/Desktop/ConsGen2018/Lab8, or wherever you chose to save the folder), then use the source() command to load the functions:

source(“scripts/coalsim\_functions.R”)

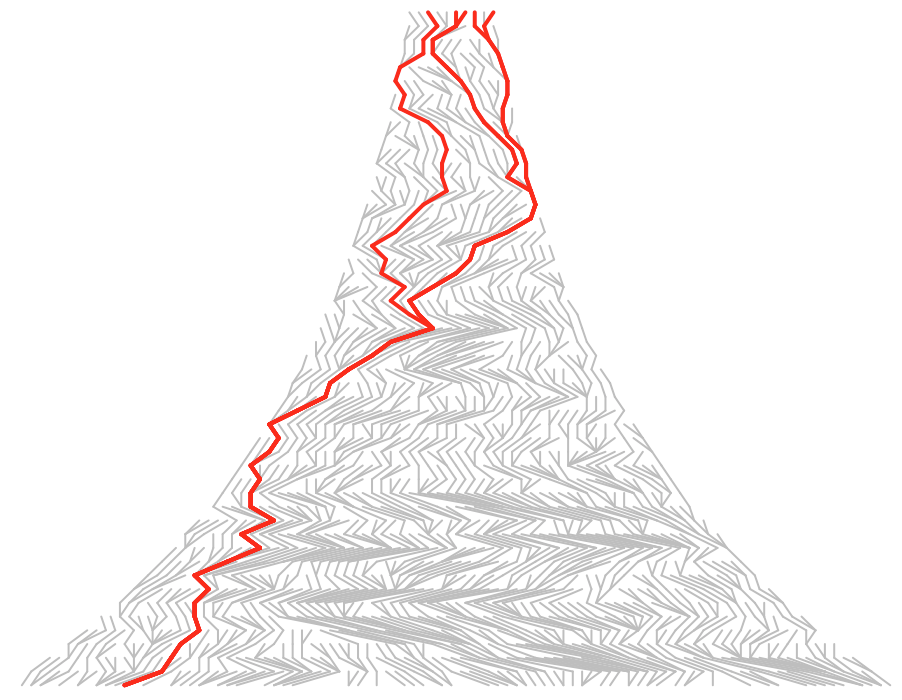
First, I want you to use the sim.coalescent() function to explore how coalescent histories are intrinsically tied to population size fluctuation. First, we will simulate the simple case of stable Ne over time:

sim.coalescent(n=5, time=200, N.0=100, N.final=100, col.lin=”grey”, col.coal=”red”)

This simulates coalescent of 5 lineages (shown in red) over 200 generations, with a start (N.0) and final (N.final) effective population size of 100. The resulting plot has time on the Y axis, with the top representing present day:

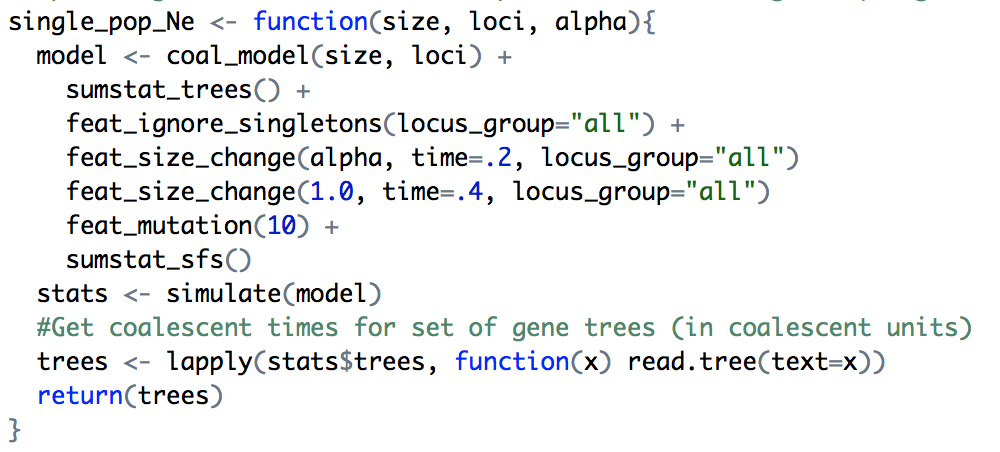


Alter the starting and ending population sizes (N.0, and N.final) to observe the effects as populations expand or contract. For example, a bottleneck would look like this:

****

**Q: Generally, what effect does population contraction or expansion have on simulated times to coalescence?** (Note: I’m not collecting answers today, but at least take a moment to think about it)

Each of the above commands has simulated a single locus. Next, we will consider coalescent variation *among* loci sharing the same demographic scenario. To do this, we will use the following function (which has already been loaded for you, assuming you completed the source() command above):



This function simulates a single population with size change at fixed time points, using the R package **coala**. Parameters specify the number of lineages to sample from the simulated population, the number of loci to simulate, and ‘alpha’, the scaling parameter for population growth or decline. An alpha value of 0.1 would indicate a population decline down to 10% of the initial size, whereas an alpha value of 10 would indicate 10X population size growth. For the purposes of this demonstration, I have fixed the population size change as instantaneous as time *t*=0.2, with a return to initial size at time *t*=0.4. You should be able to find where I have done this in the above function.

First, let’s simulate a set of **gene genealogies** under a model of stable population size. We will simulate N=15 individuals, at 6 loci:

trees<-single\_pop\_Ne(15,6,1.0)

The returned variable, ‘trees’, represents a set of phylogenies (1 for each of 100 genes). To plot them, let’s tell R to plot all six trees, then use a **for loop** to iteratively plot each one (note: it’s ok if you aren’t super familiar with R. Just copy/paste this command and you should be fine).

par(mfrow=c(2,3))

for (tree in trees){

plot(tree,show.tip.label=F, edge.width=2, direction="upwards")

}

You should see that most trees are top-heavy, meaning that most coalescence happens in the recent past. Next, let’s simulate a larger set of trees, and look at how coalescence is distributed more directly (note that 3 separate commands are shown here):

par(mfrow=c(1,1))

stable<-single\_pop\_Ne(15,100,1.0)

single\_pop\_mrca\_plot(stable, “Stable Ne”)

Here, we’ve simulated a larger set of trees (100 loci), and used the single\_pop\_mrca\_plot() function to extract coalescent times (node ages) from the resulting set of trees, and plotted a density of these. The coalescent time distribution for each locus is plotted in gray, and the overall trend in red. The x-axis is in “Coalescent Units”, each of which is equal to 2\*Ne for a diploid population. Note that most coalescence occurs very rapidly, around only 0.5 to 0.1 ‘units’ of time from the present.

We will next observe the effects of different demographic scenarios by using the alpha parameter to scale the population at a fixed time. Create 5 additional datasets, using varying degrees of population growth or decline, like so:

m1 <- single\_pop\_Ne(15,100,.5)

m2 <- single\_pop\_Ne(15,100,.1)

m3 <- single\_pop\_Ne(15,100,.01)

m4 <- single\_pop\_Ne(15,100,5)

m5 <- single\_pop\_Ne(15,100,10)

Now, plot them all next to the stable population model:

frame()

par(mfrow=c(3,2))

single\_pop\_mrca\_plot(stable, "Stable Ne")

single\_pop\_mrca\_plot(m1, "Bottleneck, a=.5")

single\_pop\_mrca\_plot(m2, "Bottleneck, a=.1")

single\_pop\_mrca\_plot(m3, "Bottleneck, a=.01")

single\_pop\_mrca\_plot(m4, "Expansion, a=5")

single\_pop\_mrca\_plot(m5, "Expansion, a=10")

Remember, I constrained population size change to the time between *t*=0.2 and *t*=0.4, so pay attention to increases or decreases in coalescence especially in that time frame.

**Q: What sort of bias does a population bottleneck cause in the distribution of coalescent times across loci? What about expansion?**

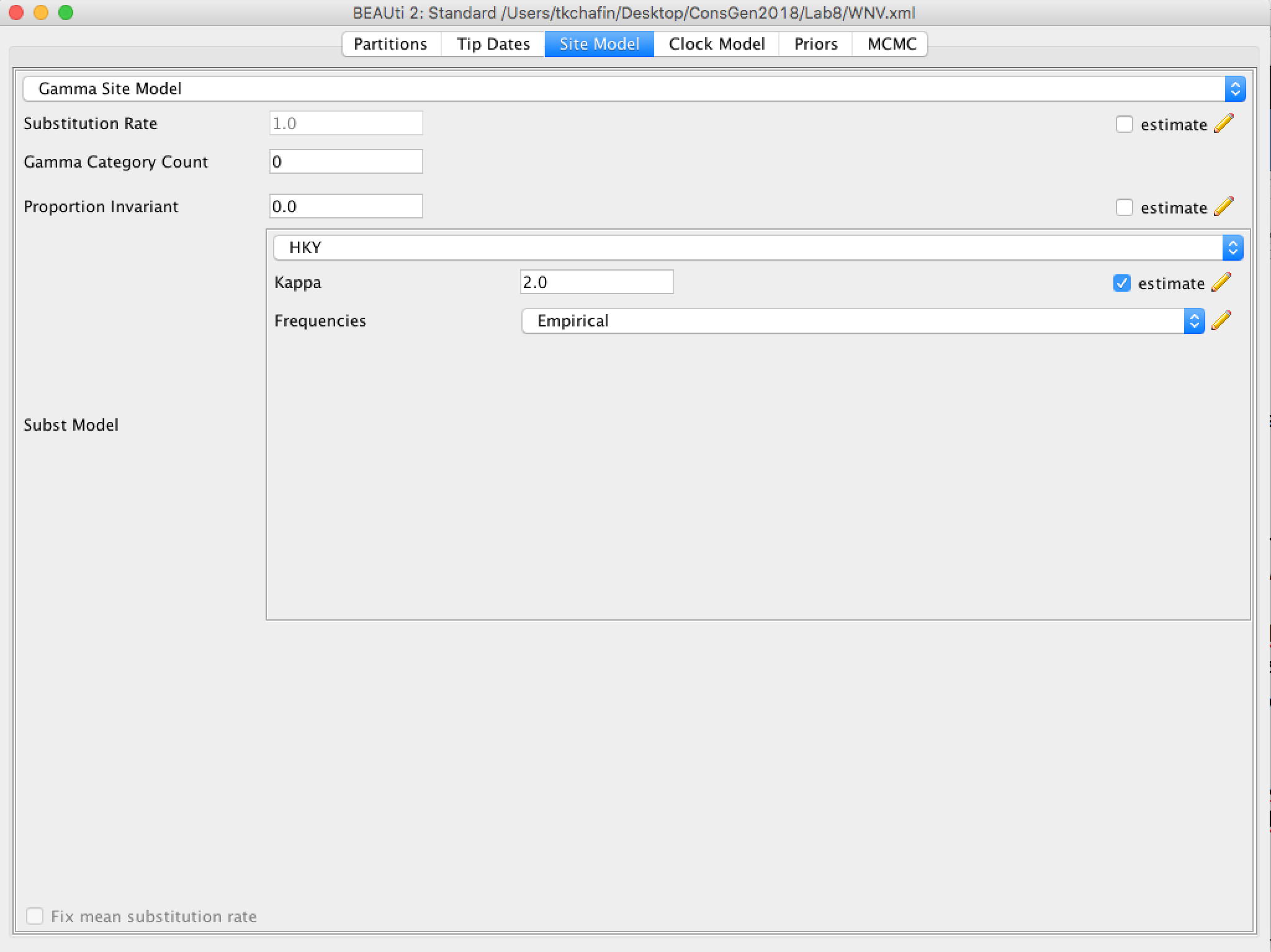
Part II – Extended Bayesian Skyline Plots (EBSP) in BEAST

To infer fluctuations in ancestral population sizes, we will be using a popular software suite for phylogenetic/phylogeographic analyses called BEAST (freely available here: <http://www.beast2.org/>).

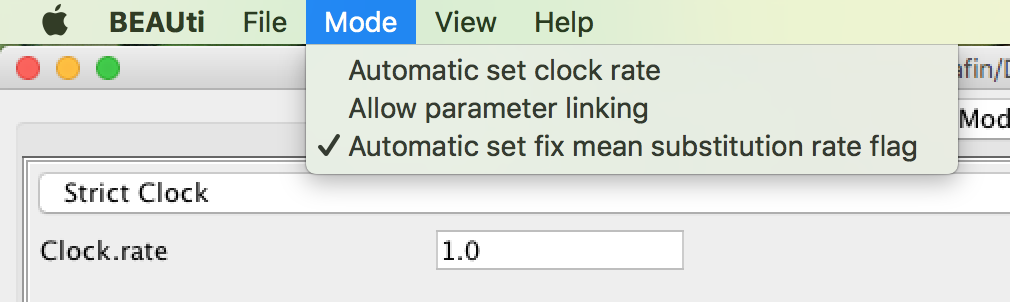
For this tutorial, we will be modelling population expansion in the Westnile Virus invasion across North America, from 104 complete WNV genomes sequenced by Pybus *et al* (2012).

To run BEAST, we first have to create an input file in a format called XML (eXtensible Markup Language), which will hold our dataset, as well as our prior specifications (since this is a Bayesian analysis), and any other parameters specified to the program. Fortunately, BEAST is distributed with a graphical interface for creating these files, called Beauti.

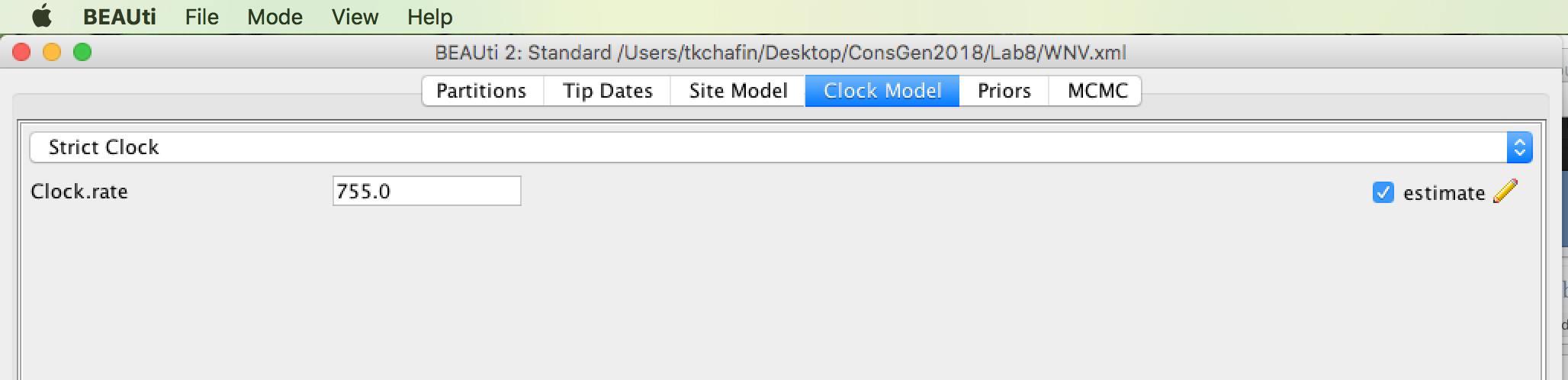
1. Open **BEAUTti** (Applications -> BEAST 2.4.3 -> BEAUti) and load the file “WNV.fasta” by going to the file menu and selectin “Import Alignment”. Make sure to tell BEAUti that this is a DNA sequence, when prompted.
2. After loading the alignment, we will first configure the nucleotide substitution model. Click on the SITE MODEL pane. Here we will choose how we think characters evolve at each aligned nucleotide position. Don’t just pick the most complex one.
   1. Usually, we could implement some form of model testing to make a more informed decision here, but for the purposes of this lab we will just select the HKY substitution model, but with ‘Frequencies’ set to Empirical:



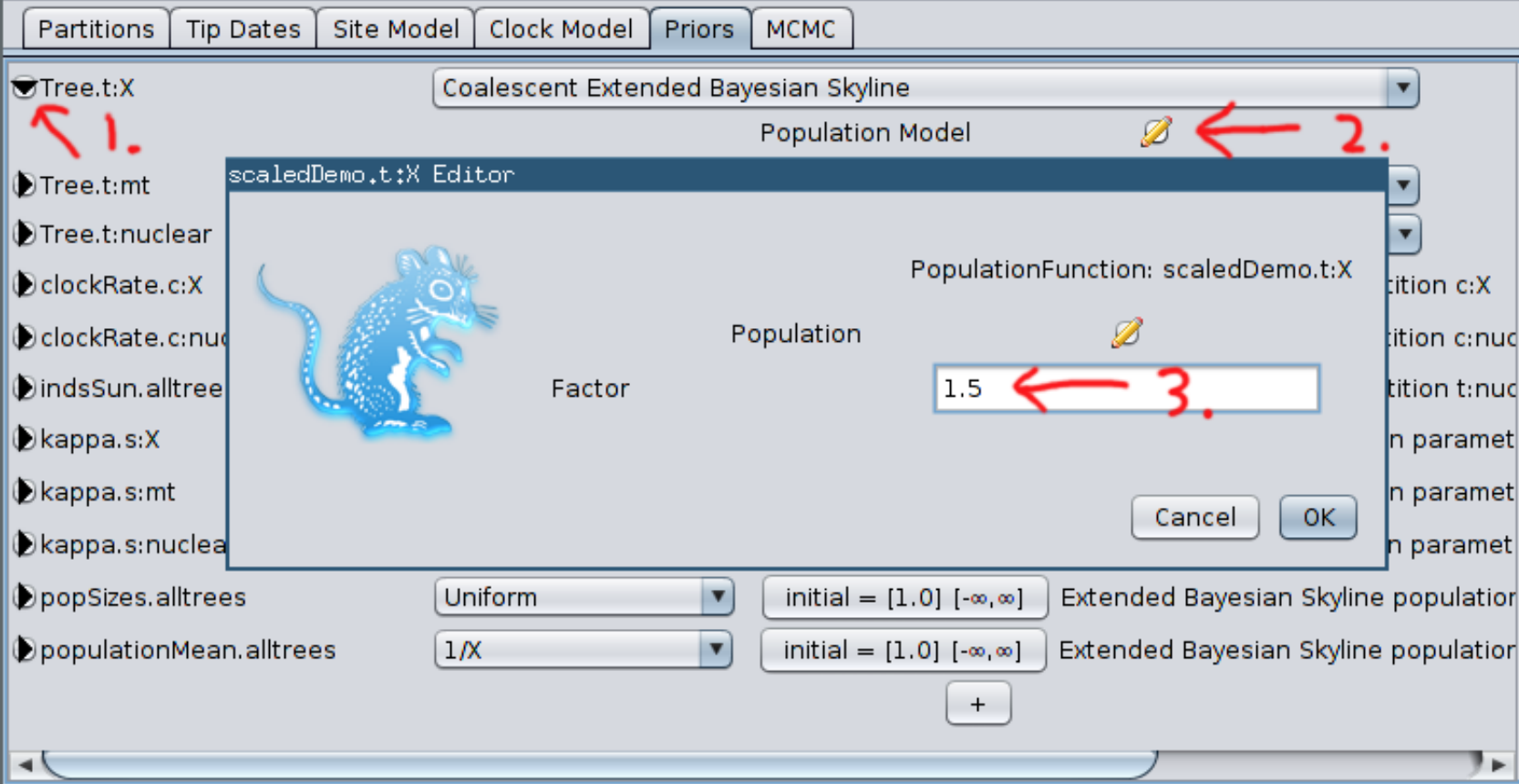
1. We then use the CLOCK MODEL pane to set a reference clock rate. This would be used to scale the resulting analysis according to a prior belief of how many substitutions occur per year. First, we must open the ‘Mode’ menu (next to ‘File’ and ‘View’) and **deselect** ‘Automatic set clock rate’.



1. Next, we choose a clock rate. This rate will correspond to the number of substitutions per million years (e.g. rate=0.05 means 0.05 substitutions per site per million years). By selecting ‘Estimate’, we would be able to provide a prior distribution to communicate our certainty about this value. For the purposes of today’s lab, we will leave it as default (rate=1.0)



1. Now we can finally set our priors. There’s always a little bit of voodoo here, and we’ll be making a lot of assumptions. In a real analysis, you may have more or less information for some priors, and would specify them accordingly. You also should try replicating your analysis with different prior specifications to assess the sensitivity of the posterior to your priors. Weak data will be easily overridden by the priors.
   1. Select the PRIORS pane. First (next to Tree.t:WNV), select ‘Coalescent Extended Bayesian Skyline’.
   2. Once this is complete, click the arrow to the left of Tree.t:WNV. Click the pencil icon next to the ‘PopulationModel’ text that appears:



* 1. This is where you would account for the ploidy of your data. For example, a diploid gene would be scaled at a factor of 2.0, because there are 2.0 copies in the population per individual. A mitochondrial gene would be scaled to 0.5, because only female mtDNA contributes to effective population size. Here, we will leave it as 1.0 (I just wanted you to see the setting ☺).
  2. Another prior which is very important is the population size distribution (populationMean.alltrees). The 1/*x* prior is generally a sensible option (and is selected by default), but may not be in all cases. Here, we’ll leave it as-is.

1. Finally, navigate to the MCMC pane. This is where we will set the **chain length for the MCMC**, as well as the **burn-in** and **sampling rate**. Refer back to Lab 7 if you need definitions for those. Set the chain length to 1 million, with a burn-in of 100,000 and sampling every 1000 iterations.
   1. **Save** your file using File->Save As, and name it “WNV.xml”, in the same directory as the rest of your Lab 8 files.
2. Close BEAUti. Run the analysis by opening BEAST (Applications->BEAST 2.4.3->BEAST), and select “Overwrite log files” from the dropdown box. Click **Run**. The analysis should take less than a few minutes.

After the analysis has completed, we would typically inspect the results to assess the MCMC for convergence, proper mixing, and to better determine our burn-in. We will learn how to do this in a later lab, using the program **Tracer**. For now, just take my word that your MCMC will have very poor mixing. Sampling of substitution, population parameters, and rates would be found in the output “WNV.log” produced by BEAST. Other outputs include WNV.trees, which includes the sampled trees, and the “EBSP.log” file which will be used to plot our results.

The result likely has very wide credibility intervals because the chain was not run long enough to achieve convergence. **We will need to run the analysis much longer (tens of millions of iterations) to achieve good mixing.**

Just kidding. I’ve already done that for you. Unzip the wnv\_ebsp.tar.gz archive to find results for a longer run with 20 million iterations, 5 million burn-in, and thinned sampling to every 5000 iterations. Then go back to your R console (or R studio). After source’ing the plotEBSR.R file in the Lab8/scripts directory, you can plot both your short and longer runs using the following command (note file names/paths may need to be changed):

plotEBSP(“EBSP.log”, useHPD=F, log=”y”)

plotEBSP(“long\_run/EBSP.log”, useHPD=F, log=”y”)

The resulting plot should be pretty straightforward to read. The X-axis is in number of substitutions (NOT scaled to millions of years, unless a clock rate was specified), and the y-axis is estimated Ne on a log scale.

Pybus, O. G., *et al*. 2012. Unifying the spatial epidemiology and molecular evolution of emerging epidemics. *PNAS.* 109:15066-15071.

May, F. J., *et al*. 2011. Phylogeography of West Nile Virus: from the Cradle of Evolution in Africa to Eurasia, Australia, and the Americas. *J. Virology*. 85(6): 2964-2974.