Extended Bayesian Skyline Plot tutorial

Joseph Heled[1] jheled@gmail.com

October 11, 2010

This short practical explains how to set up an *Extended Bayesian Skyline Plot* (EBSP) analysis in BEAST[1], and how to generate some EBSP plots.

To make the most of this tutorial you should follow the steps described here on your system. Reading the article first is not a bad idea either[2]. Use your own data if you are comfortable with BEAST, otherwise use the provided examples. The next section uses the data in the '`mystery-mammal`' directory, 3 genes from a small mammal: one nuclear, one mitochondrial, and one from the X chromosome.

This tutorial is based on BEAUti version 1.6.

My sincere apologies for any harm done by the figures to your aesthetic sensibilities. Nobody, the Java team included, cares much for my preferred platform, Linux.

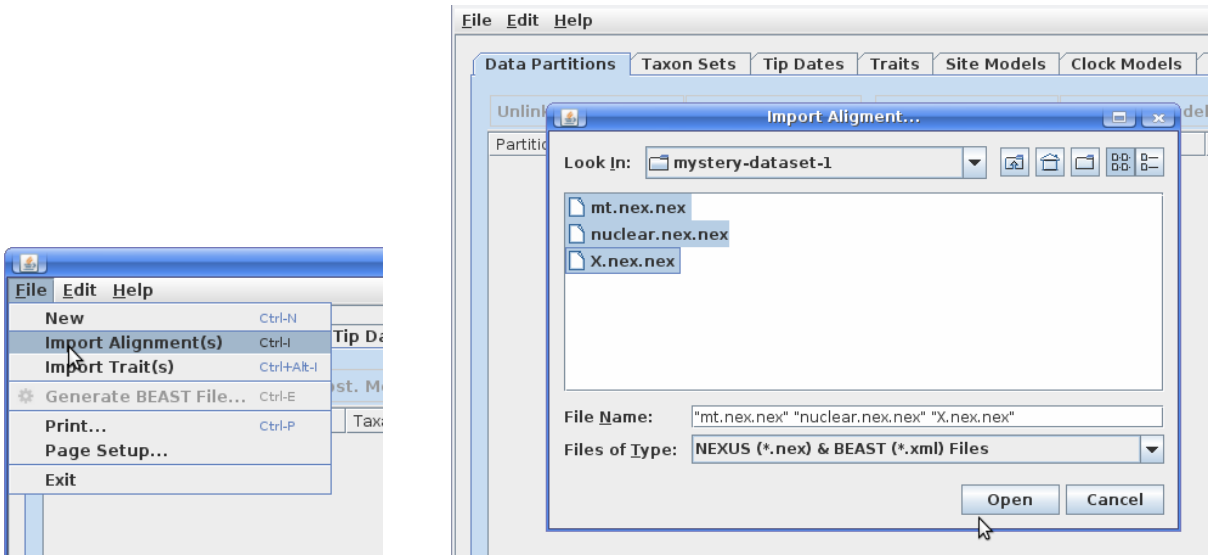
1 Setting Up the Analysis

Loading the Data

The EBSP is a multi-locus method, so the step first involves loading all loci/genes into BEAUti. The simplest way to do so is to prepare one NEXUS file for each alignment. Taxa names in each alignment have to be unique, but duplicates across alignments are fine.

Start BEAUti and click File|Import, or hold the control key and press 'I' (`C-I` in short). Navigate to the data directory, select all the files you wish included in the analysis and click 'Open' (Figure 1).

Click 'Yes' in the pop-up asking you to confirm using different taxa in each partition.



(a) (b)

Figure 1: Loading multiple alignments into BEAUti

Unlink partitions

By default BEAUti links the partitions trees, clock and substitution models. This is fine if the genes are known to evolve in tandem, but here we shall unlink them. Select all partitions (`C-A`) and click on 'Unlink Subst. Models', 'Unlink Clock Models' and 'Unlink Trees' (Figure 2).

Site Models

Move to the 'Site Models' section. Here we choose, for every gene, the evolutionary models specifying how characters evolve at each site (one aligned position). Do not automatically choose the most general models possible (GTR, Gamma + Invariant Sites, Partitioned Codon positions etc.). Please remember that the more general models have more parameters, and this may result in longer runs and slower mixing. This is the time to display your superior knowledge of the data! Sometimes performing an exploratory run(s) on a single loci can help, especially when the simpler model is a special case of a general one. Sometimes the simpler model is represented by particular parameter values, and checking if the credible interval(s) contain those specific values can be a factor in the decision.

In this particular case leave the defaults, but change the 'Base frequencies' to `Empirical`. Repeat 3 times for each loci (Figure 3). Please note that I know absolutely nothing about this particular data, and it is entirely possible using the GTR or partitioning is the better choice.

Clock Models

Now move to the 'Clock Models' section. Here you choose the molecular clock model for each gene. The same considerations as in the site models apply: stay with the strict clock unless you know otherwise.

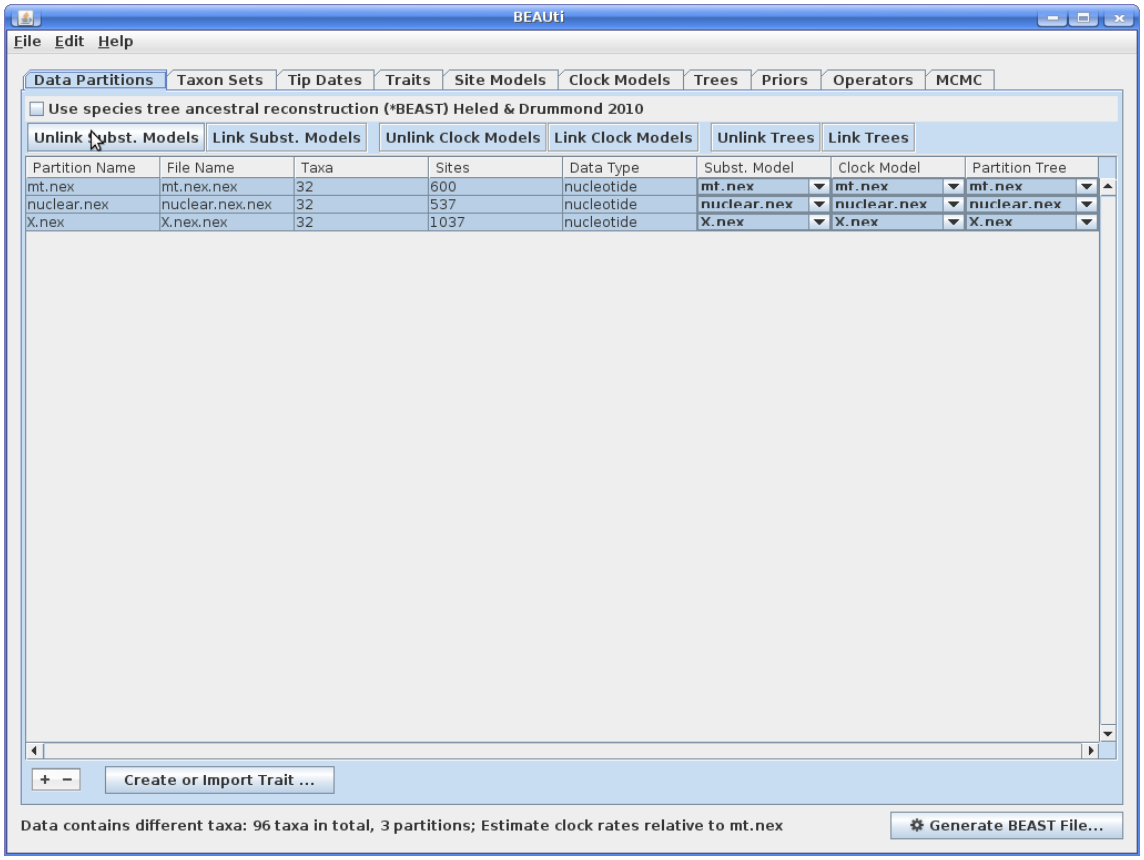


Figure 2: Unlinking clock and substitution models.

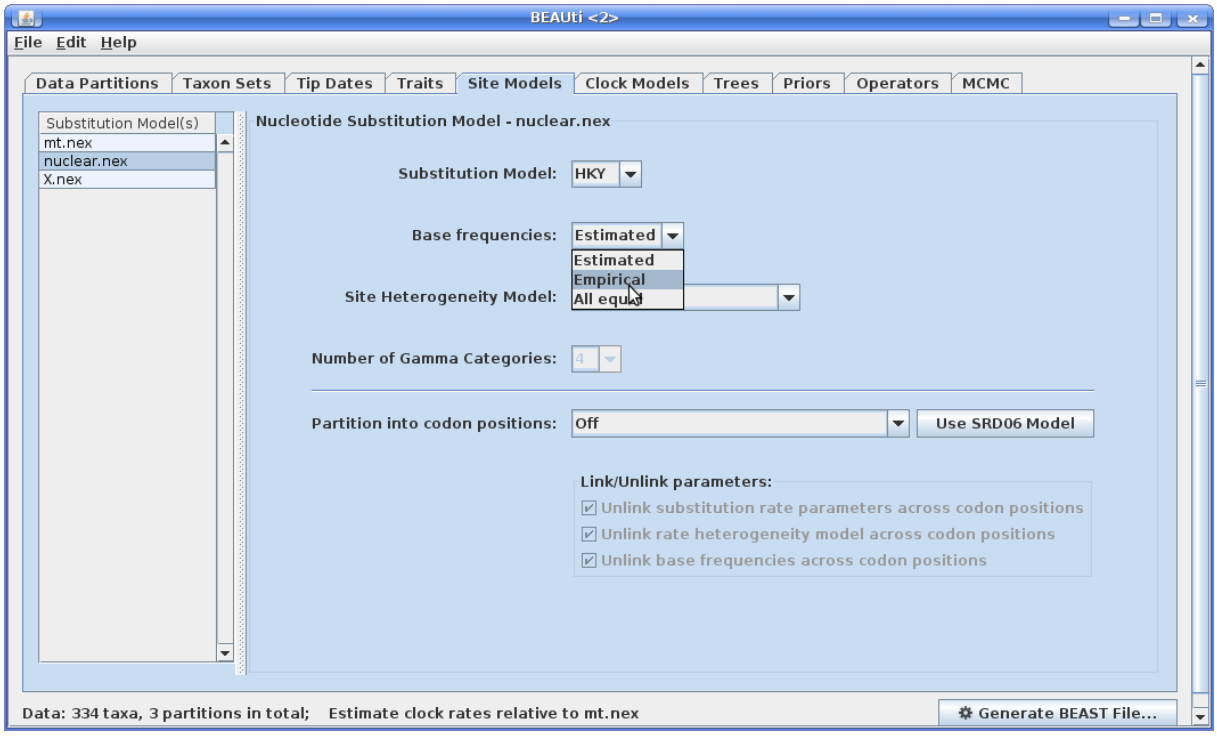


Figure 3: Setting the site model.

You may have noticed that exactly two of the three genes are marked as 'ESTIMATE'. In general we can't infer the absolute rate from contemporaneous data (all samples collected at the same time), but we can infer relative rates. The unmarked loci is the reference rate, which is by default set to one. Other rates will be relative to the reference: a rate of 2 (with the reference being 1) means "evolving twice as fast as the reference".

In general it is best to pick the most stable reference. Here this would be the mtDNA, since it evolves faster, and so the sequences in that loci will be more divergent and contain more information. Insure that the rate of 'mt.nex' is unchecked and the other two are checked.

The units of the time axis for the trees and population size is the same as the reference rate. When it equals 1, time is measured in substitutions. When we have an estimate of the reference, it may be more convenient to plug it in and have time in years. Since this is a mammal, I will arbitrarily pick a rate of 0.05 per million years[3]. Double-Click on the 1.0 rate of the mt row and enter 0.05 – and times from the analysis will be in millions of years (Figure 4).

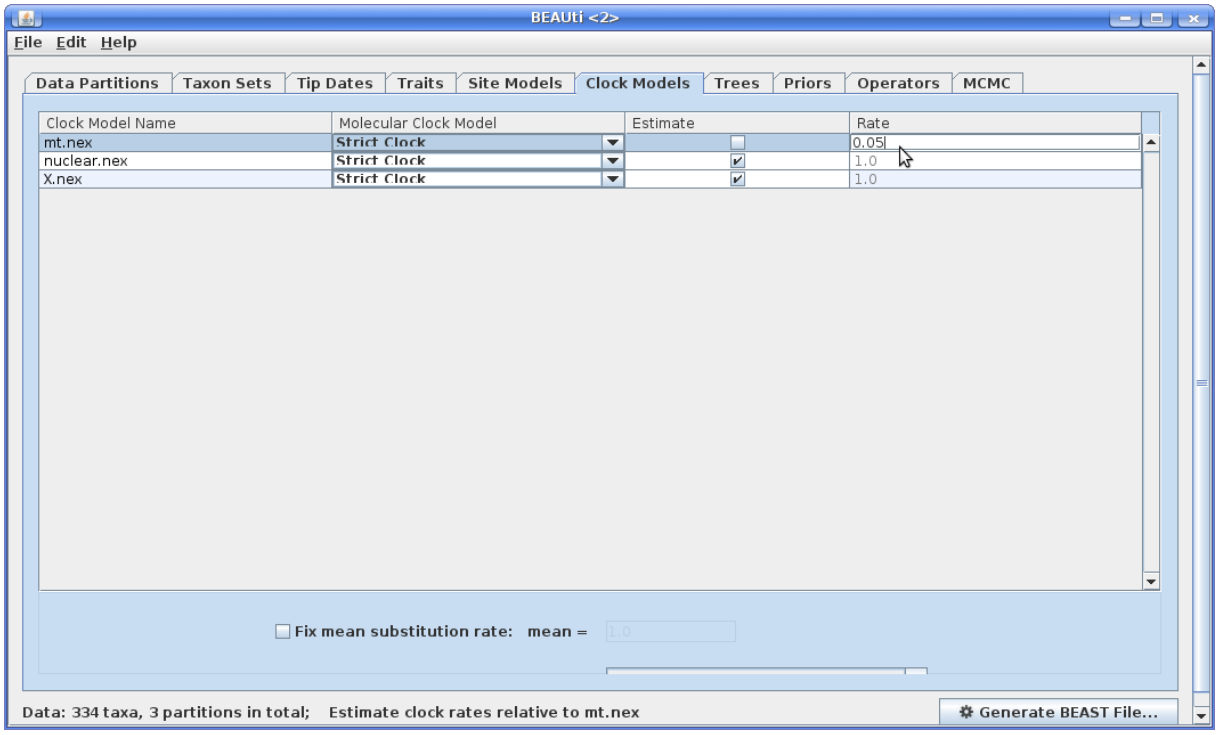


Figure 4: Setting the mtDNA rate

To make a smoother start you should change the other starting rates – say to 0.005. Unless this bug is fixed by the time you read this, you have to uncheck the 'ESTIMATE' box first, change the value, then check it again.

Note that we set the reference rate to a known fixed value, but that can be relaxed too. It is possible to let the rate vary (by checking it's 'Estimate' box as well), and place a prior on the rate parameter in the priors section (which is `mt.nex.clock.rate` in this specific case). You have to realize that the rate will not be estimated – there is no data here to make that possible – it will simply follow the prior. But the uncertainty regarding the exact value will be reflected in all other estimates.

Trees Section

On to the 'Trees' section. From the 'Tree Prior' drop-down menu choose the `Coalescent: Extended Bayesian Skyline`. Note the magical appearance of new options.

First we have to set the right 'Ploidy Type' for each gene. Genes at specific loci collected from organisms may exist in multiple copies. But you probably know the details much better than I do. The practical implication for us is that genes sampled from an organism may have different population sizes. The 'Ploidy' setting takes care of those differences. Choose the `mitochondrial` option from the drop-down menu for `'mt.nex'`, `X` for `'X.nex'`, and leave `'nuclear.nex'` in its default `autosomal nuclear` (Figure 5).

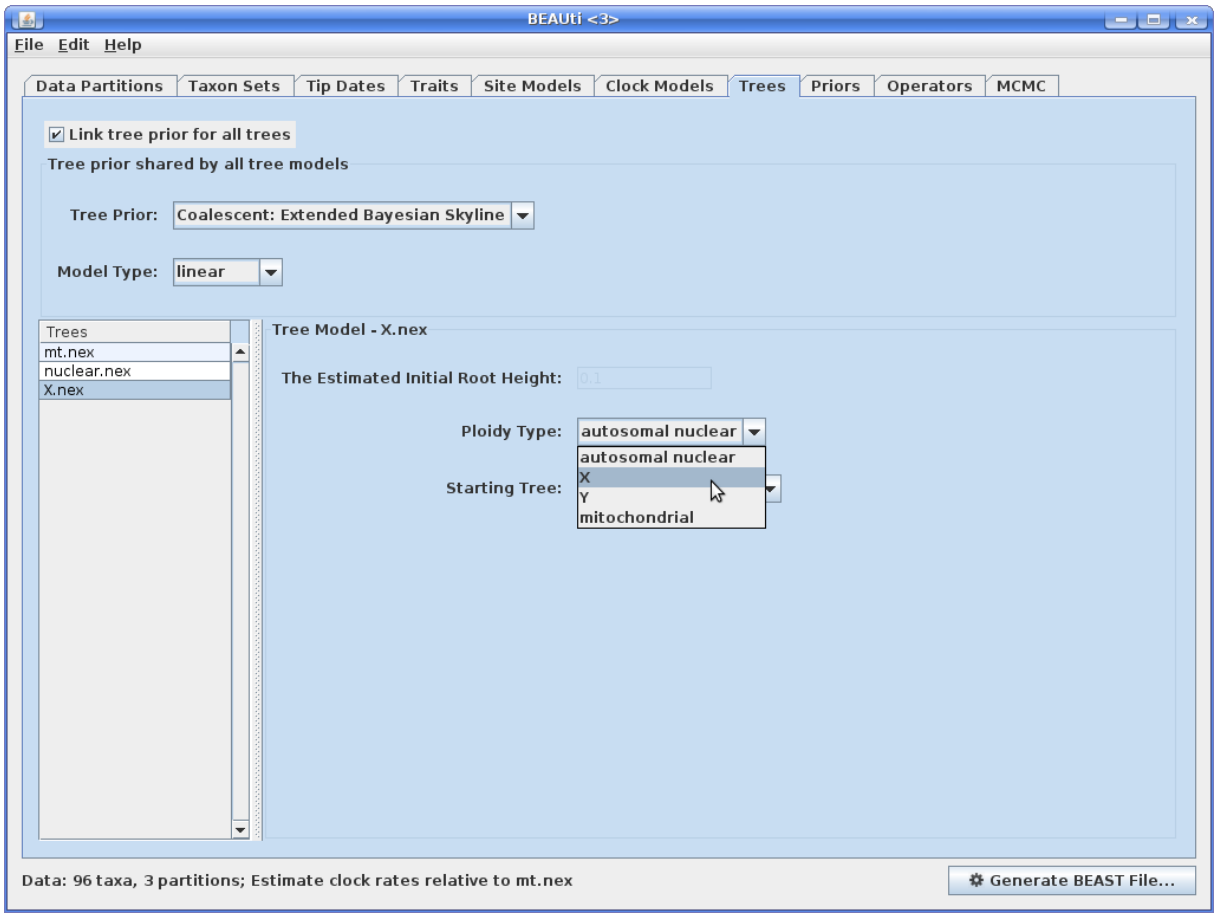


Figure 5: Setting genes ploidy.

The other option you may consider changing is the 'Model Type'. Currently there are two options, `linear` (the default) and `stepwise`. With linear, the population size function is piecewise-linear, that is made of a series of connected straight line segments. With stepwise, it is made of a series of unconnected segments, each parallel to the X axis. I find the linear model more natural – real life population size is continuous – but in some cases the stepwise model may be more appropriate. One case that comes to mind is serial data, where the samples are taken at several different time points, and it may make sense not to force continuity between sample points.

Priors Section

Click to the 'PRIORS' section. BEAUti 1.6 choose not to provide a prior for the clock rates (in my book, uniform on $[0, \infty]$ for a positive parameter does not deserve this designation). There are many reasonable choices. We shall set it to uniform on the range $[0, 0.05 \times 20]$. Click on the prior button to the right of `nuclear.nex.clock.rate`, and fill in the values (Figure 6). Repeat for `X.nex.clock.rate`.

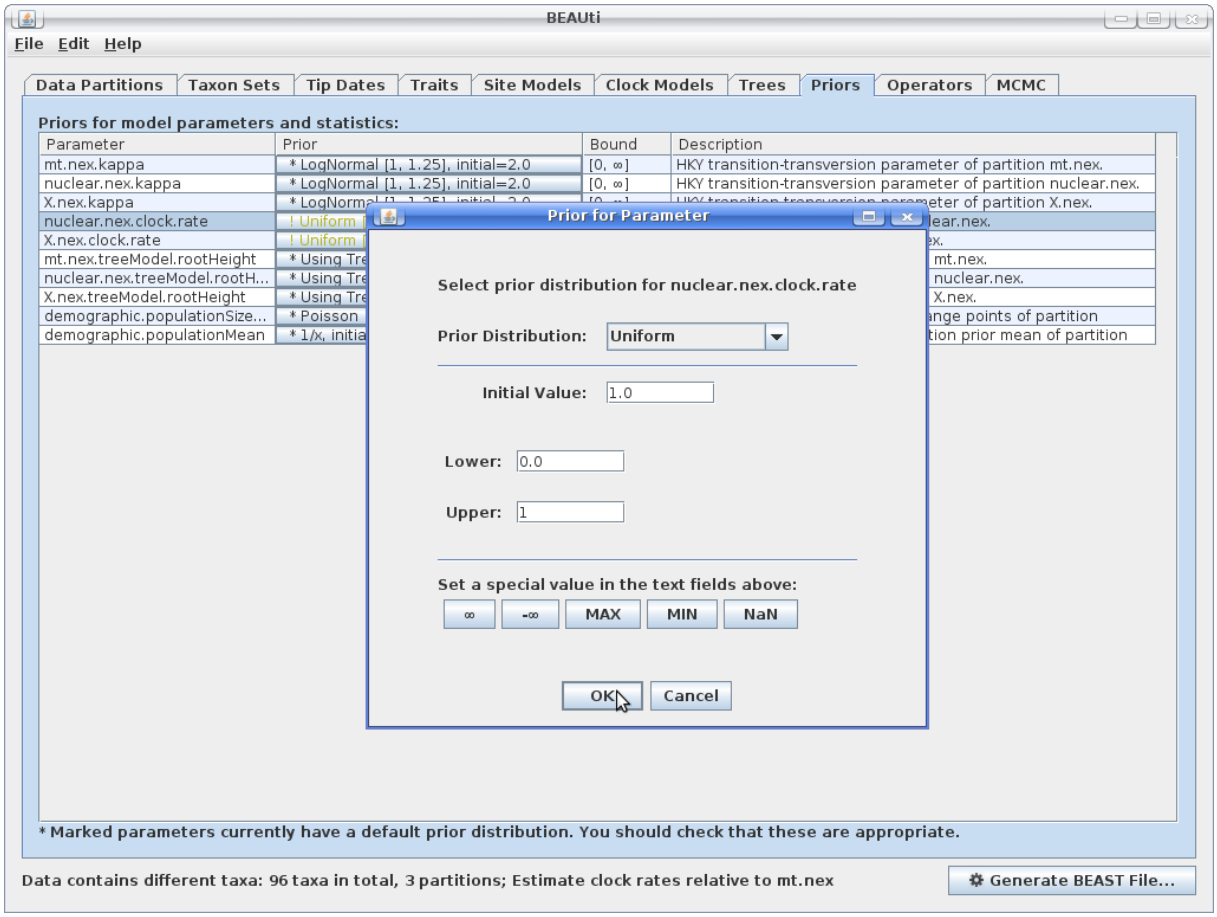


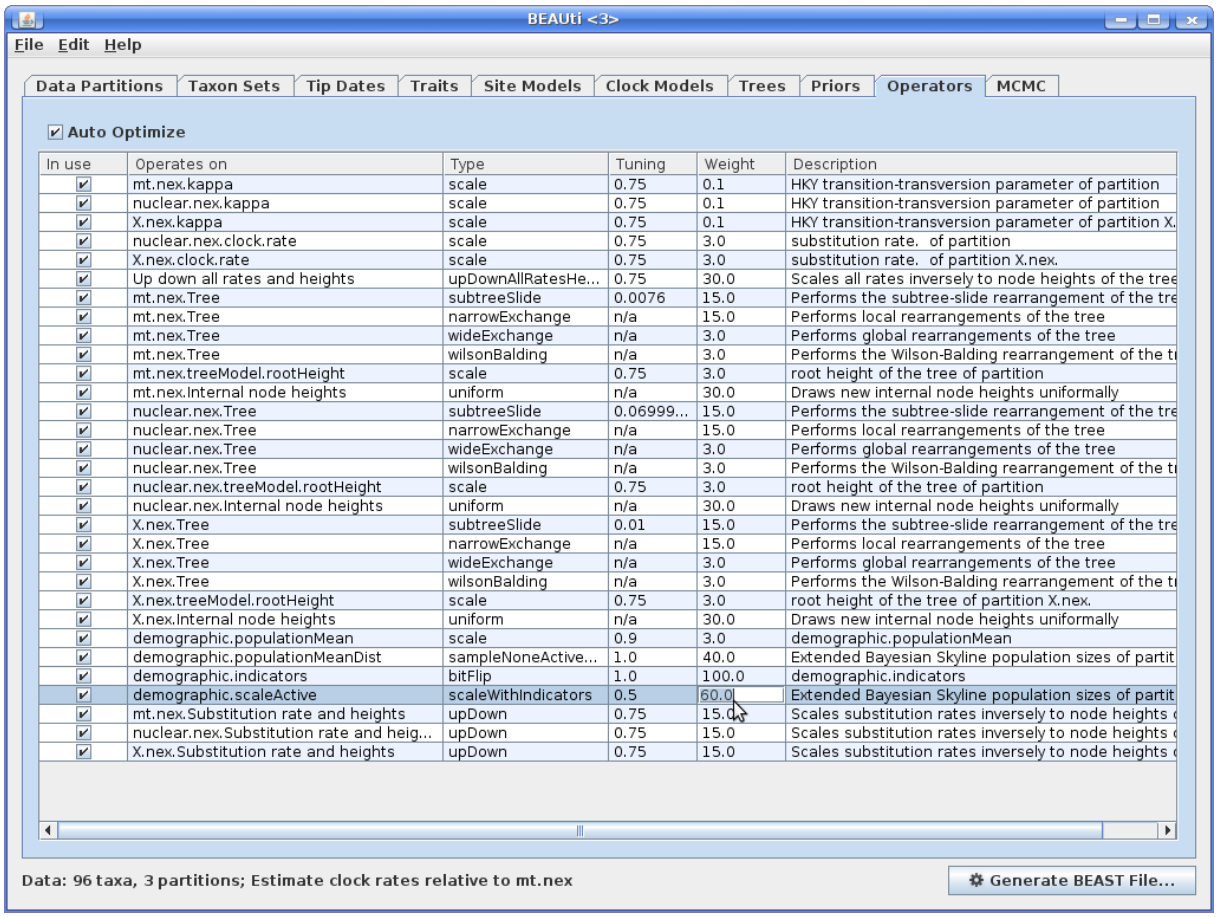
Figure 6: Setting rates prior.

Operators Section

Off we go to the 'OPERATORS' section. Typically we skip this section entirely, but no such luck this time. We need to take care of two issues. First, the default operator weights generated by BEAUti for the EBSP are terrible. True, we do not know how to assign weights to get the best mixing from a chain. Comparing chains divergence is devilishly complicated. Still, there is some scope for intuition supported by some personal experience (yes, which can prove faulty later).

The weights for each gene tree sum up to 69, which gives about 210 for the 3 genes. The weights for the EBSP operators are too low in comparison. We shall change the weights for `demographic.populationMeanDist`, `demographic.indicators` and `demographic.scaleActive` to 40, 100 and 60, respectively. Double-Click on the weight and enter the value (Figure 7).

Also up the 3 'rate and heights' operators at the end of the list from 3 to 15. And while it is true that a low weight for 'kappa' is fine, 0.1 is on the ridiculous side. Change them to 2.



In use	Operates on	Type	Tuning	Weight	Description
☑	mt.nex.kappa	scale	0.75	0.1	HKY transition-transversion parameter of partition
☑	nuclear.nex.kappa	scale	0.75	0.1	HKY transition-transversion parameter of partition
☑	X.nex.kappa	scale	0.75	0.1	HKY transition-transversion parameter of partition X.
☑	nuclear.nex.clock.rate	scale	0.75	3.0	substitution rate. of partition
☑	X.nex.clock.rate	scale	0.75	3.0	substitution rate. of partition X.nex.
☑	Up down all rates and heights	upDownAllRatesHe...	0.75	30.0	Scales all rates inversely to node heights of the tree
☑	mt.nex.Tree	subtreeSlide	0.0076	15.0	Performs the subtree-slide rearrangement of the tre
☑	mt.nex.Tree	narrowExchange	n/a	15.0	Performs local rearrangements of the tree
☑	mt.nex.Tree	wideExchange	n/a	3.0	Performs global rearrangements of the tree
☑	mt.nex.Tree	wilsonBalding	n/a	3.0	Performs the Wilson-Balding rearrangement of the t
☑	mt.nex.treeModel.rootHeight	scale	0.75	3.0	root height of the tree of partition
☑	mt.nex.Internal node heights	uniform	n/a	30.0	Draws new internal node heights uniformally
☑	nuclear.nex.Tree	subtreeSlide	0.06999...	15.0	Performs the subtree-slide rearrangement of the tre
☑	nuclear.nex.Tree	narrowExchange	n/a	15.0	Performs local rearrangements of the tree
☑	nuclear.nex.Tree	wideExchange	n/a	3.0	Performs global rearrangements of the tree
☑	nuclear.nex.Tree	wilsonBalding	n/a	3.0	Performs the Wilson-Balding rearrangement of the t
☑	nuclear.nex.treeModel.rootHeight	scale	0.75	3.0	root height of the tree of partition
☑	nuclear.nex.Internal node heights	uniform	n/a	30.0	Draws new internal node heights uniformally
☑	X.nex.Tree	subtreeSlide	0.01	15.0	Performs the subtree-slide rearrangement of the tre
☑	X.nex.Tree	narrowExchange	n/a	15.0	Performs local rearrangements of the tree
☑	X.nex.Tree	wideExchange	n/a	3.0	Performs global rearrangements of the tree
☑	X.nex.Tree	wilsonBalding	n/a	3.0	Performs the Wilson-Balding rearrangement of the t
☑	X.nex.treeModel.rootHeight	scale	0.75	3.0	root height of the tree of partition X.nex.
☑	X.nex.Internal node heights	uniform	n/a	30.0	Draws new internal node heights uniformally
☑	demographic.populationMean	scale	0.9	3.0	demographic.populationMean
☑	demographic.populationMeanDist	sampleNoneActive...	1.0	40.0	Extended Bayesian Skyline population sizes of partit
☑	demographic.indicators	bitFlip	1.0	100.0	demographic.indicators
☑	demographic.scaleActive	scaleWithIndicators	0.5	60.0	Extended Bayesian Skyline population sizes of partit
☑	mt.nex.Substitution rate and heights	upDown	0.75	15.0	Scales substitution rates inversely to node heights
☑	nuclear.nex.Substitution rate and heig...	upDown	0.75	15.0	Scales substitution rates inversely to node heights
☑	X.nex.Substitution rate and heights	upDown	0.75	15.0	Scales substitution rates inversely to node heights

Figure 7: Setting operators weights.

The second issue is more complex. While we can run the analysis as is, the EBSP default is to have a $^1/_x$ hyper prior on the mean of the distribution of population sizes. This is a safe choice but one that typically leads to slow mixing. There are several ways to asses the magnitude of the population prior to running. Running just the reference gene (mtDNA) with a Coalescent prior and constant population size can give a good idea of the range. In here we shall fix the mean to 1. Go back to the Priors section, click on the `demographic.populationMean` prior and set the initial value to 1 (Figure 8).

Back in the Operators page, uncheck the 'In use' box of `demographic.populationMean`. Unfortunately, even this is not enough since the population mean is changed in another operator, one we wish to keep. So, **after** we save the BEAST XML file in the next section, fire up your favorite text editor and search for the line

```
<parameter idref="demographic.populationMean"/>
```

inside the up-down operator (Figure 9). **Then delete it**.

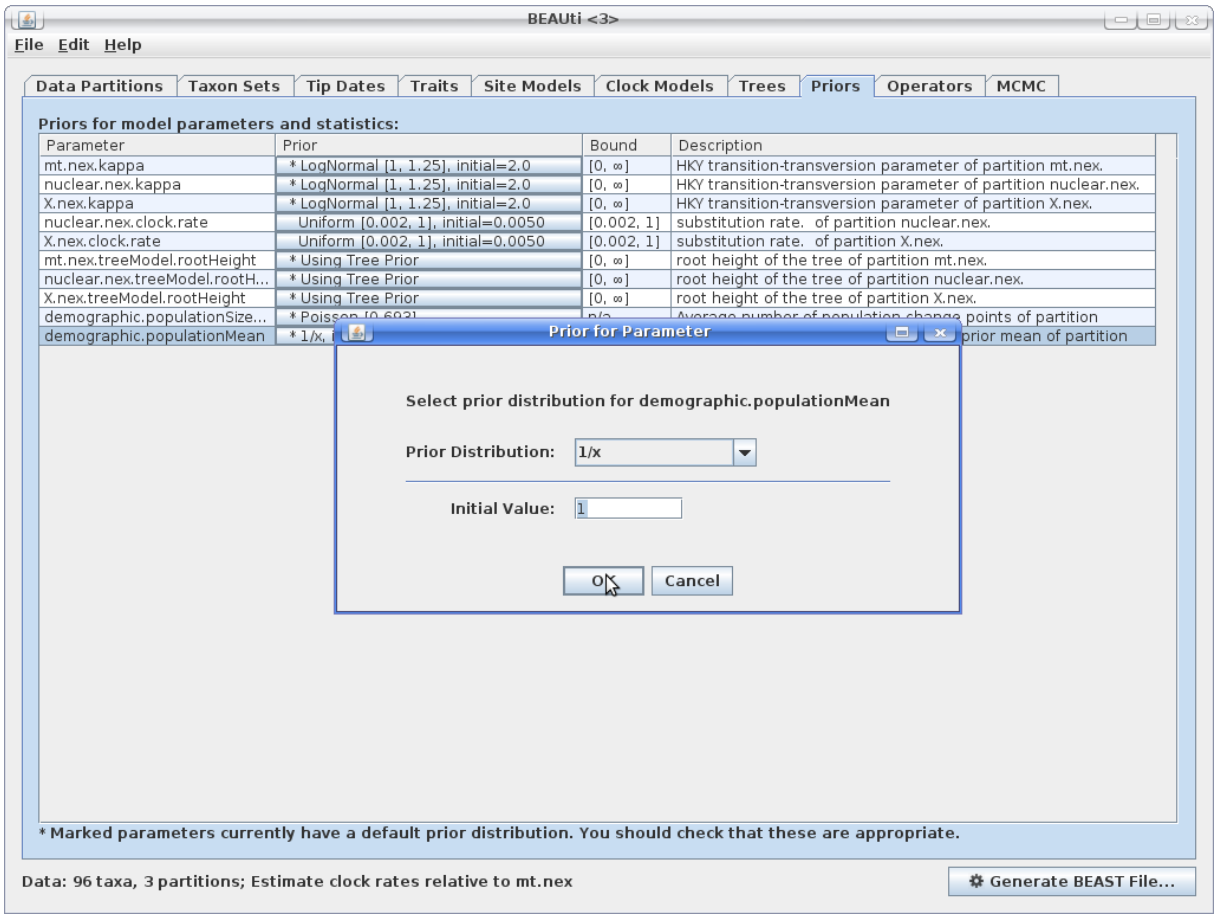


Figure 8: Set Population Mean value to 1.

```
</scaleOperator>
<scaleOperator scaleFactor="0.75" weight="3">
        <parameter idref="X.nex.clock.rate"/>
</scaleOperator>
<upDownOperator scaleFactor="0.75" weight="30">
        <up>
                <parameter idref="nuclear.nex.clock.rate"/>
                <parameter idref="X.nex.clock.rate"/>
        </up>
        <down>
                <parameter idref="demographic.populationMean"/>
                <parameter idref="demographic.popSize"/>
                <parameter idref="mt.nex.treeModel.allInternalNodeHeights"/>
                <parameter idref="nuclear.nex.treeModel.allInternalNodeHeights"/>
                <parameter idref="X.nex.treeModel.allInternalNodeHeights"/>
        </down>
</upDownOperator>
<subtreeSlide size="0.008799999999999999" gaussian="true" weight="15">
        <treeModel idref="mt.nex.treeModel"/>
</subtreeSlide>
```

Figure 9: Remove population mean from up-down operator.

MCMC Section

Final stop: 'MCMC' section. We will change the name to `mystery`, reduce the length of the chain to 5 Million, each every 5000 states and log every 2000 (Figure 10).

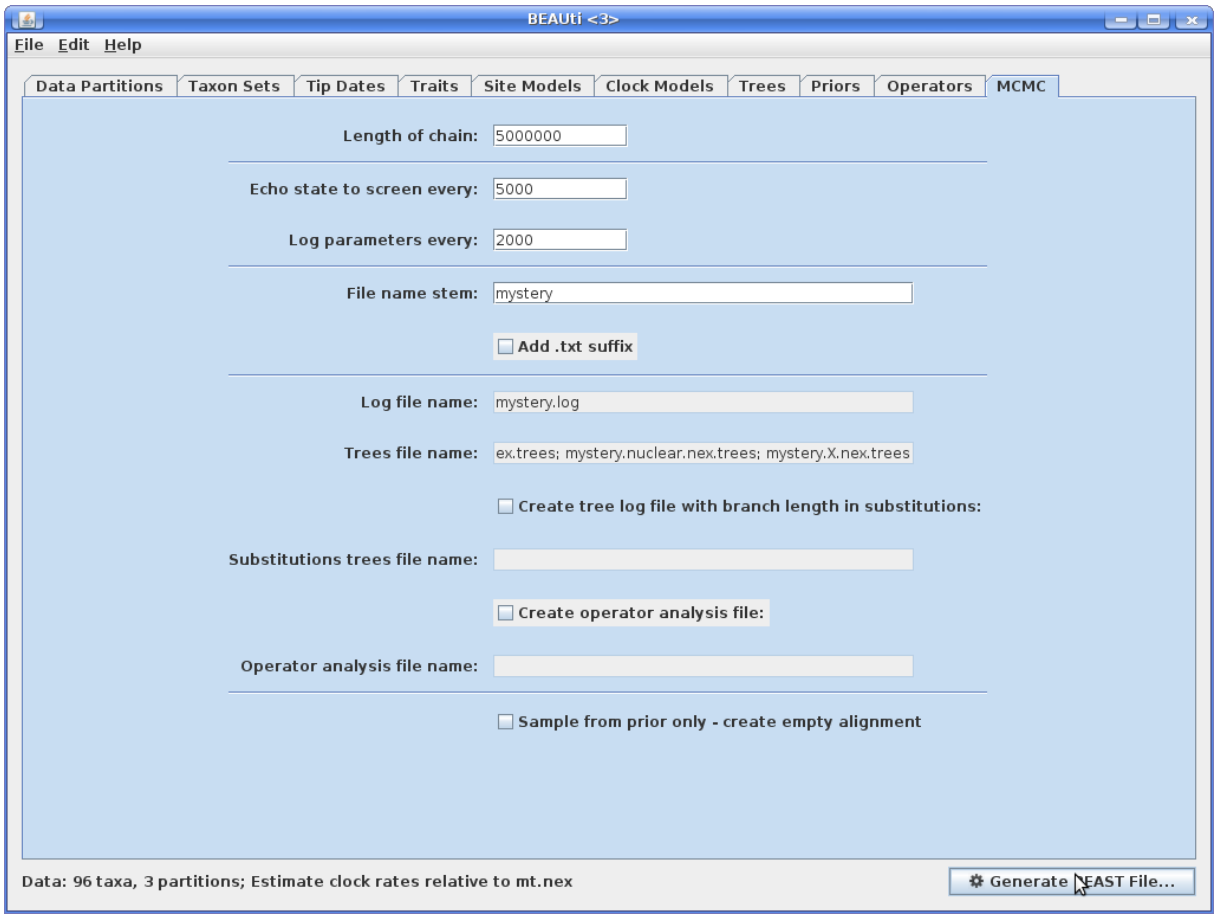


Figure 10: MCMC panel.

Note that 5M is almost certainly too short – this is just to get a fast running practical. Click the 'GENERATE BEAST FILE...' button, 'CONTINUE' on the annoying popup, and click 'SAVE'. I suggest you save the file in a new empty directory.

2 Running, Inspecting and Plotting

Run the chain using BEAST. On a properly set up BEAST and a reasonable machine this should take less than 15 minutes. Or you can skip this stage and use the output in the '`mystery-run-1`' directory.

Tracer inspection

Start up Tracer and load the '`mystery.log`' log file. I suggest you browse a little on your own before reading further.

I guess you looked at the posterior ESS (Effective Sample Size) first. The calculated value is 102, but from visual inspection I suspect it is even lower. That's fine, we knew 5M would be short (Figure 11).

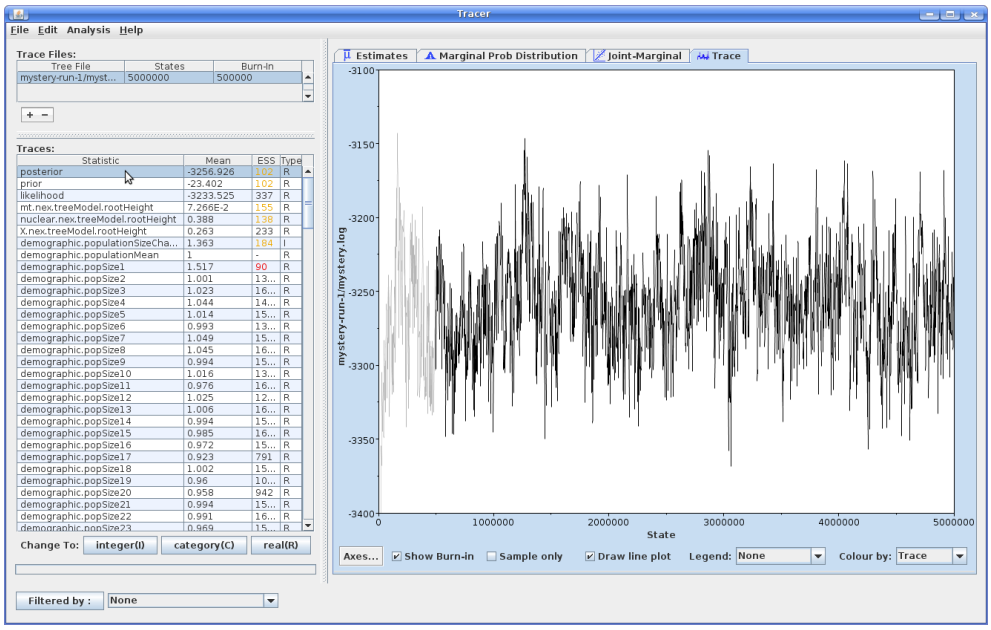


Figure 11: Tracer Panel.

The next stop for an EBSP analysis is the number of population changes (`demographic.populationSizeChanges`). We can be fairly certain in rejecting a constant population since the credible set contains 1 and 2 changes (Figure 12).

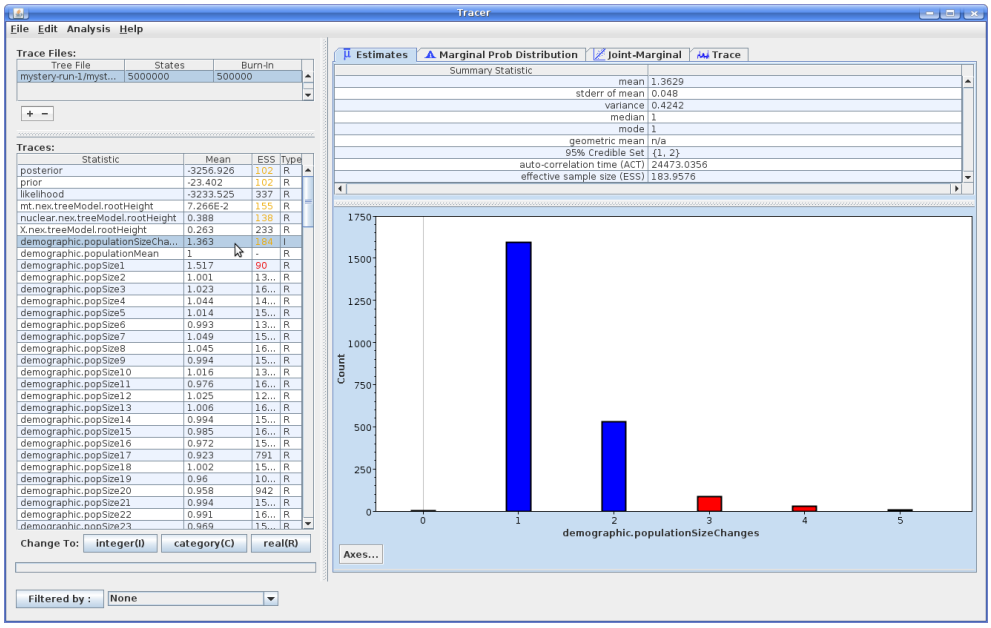


Figure 12: Tracer Panel.

For our third stop scroll all the way down. The nuclear and X clock rates came out at around 8E3 and 2E3, which seem reasonable. The kappa values look fine; not sure about the

X – it has a very wide credible interval – I am guessing there are too few substitutions to enable a good estimate.

Plotting

The EBSP analysis generated a '`mystery.csv`' file. This file contains the population size function and a plot can be made in any spreadsheet program. Alternatively you can use my own Python scripts, which requires several packages, mainly scipy and matplotlib (which includes pylab). I can't help you with the installation of python or those packages, but on Ubuntu they are a few clicks away. Assuming you have the proper setup, run the following command in the run directory:

```
../scripts/popGraphFromCSV.py --logy mystery.csv mystery1.png
```

This assumes the run directory is just below the main tutorial directory. Here is the generated plot (Figure 18):

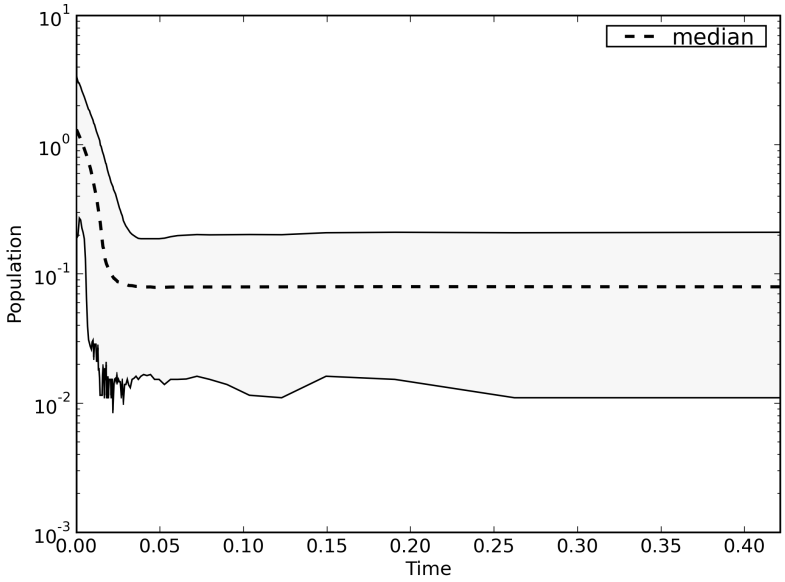


Figure 13: The EBSP plot. The median population size in the dashed line, and the 95% HPD in the lightly shaded grey. The Y axis is in logarithmic scale.

The script has various options. to see them all execute

```
../scripts/popGraphFromCSV.py  --help
```

Let us generate a close up with,

```
../scripts/popGraphFromCSV.py  --xlim 0.03 ./mystery.csv ./mystery-2.png
```

Figure 14 shows a ten-fold increase over the past 20000 years (remember our times are in Myears). If we assume a generation time of 1 Year, we get a current effective population size

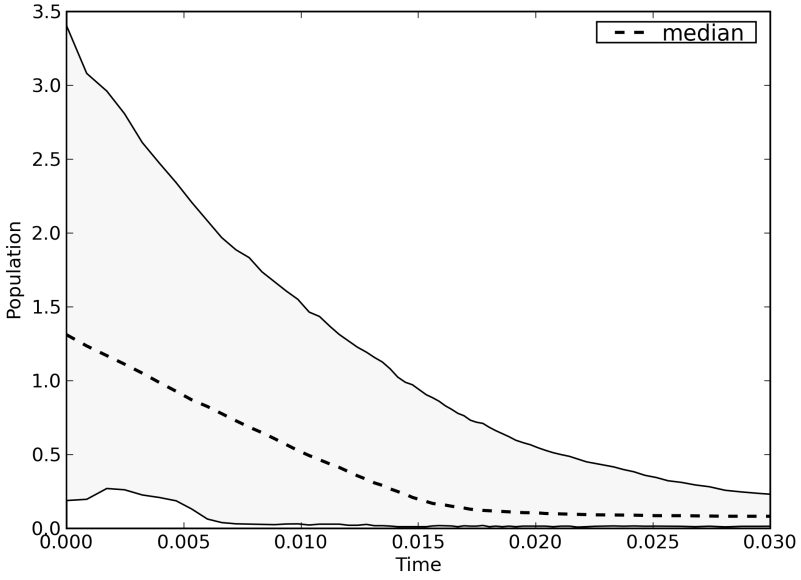


Figure 14: .

of around 1.3 million*. This is a very large number, but remember that this is for illustration purposes only. The actual numbers depend on choice of mutation rate and generation time. Ignoring this objection, let us explore the result a little more.

First note the wide credible intervals. Three genes will rarely give tight bounds when dealing with population sizes. The need for more genes when estimating effective population size can not be overemphasized. Especially when considering that I used only 32 sequences out of a about 128. For the purpose of this analysis the effort in sequencing this large number is wasted: sequencing more genes, if possible, would greatly improve the results.

To continue our exploration we would like to generate more plots, and to do so we need to generate a more detailed '.csv' file first. We will prepare a new XML file. Open the '`mystery.xml`' with a text editor and copy the last two sections, `VDAnalysis` and `CSVexport` to a new file (say '`alld.xml`'). Place them inside a '`<beast> </beast>`' section. Now make the following manual changes: add `nBins="200"` to the VDAnalysis attributes, and add those two terms at the end of the element:

```
<allDemographicsFileName>
  mystery.alld.txt
</allDemographicsFileName>
<rootheightColumn>
  nuclear.nex.treeModel.rootHeight
</rootheightColumn>
```

See '`mystery-run-1/alld.xml`' if you are confused.

The first element is a name of a file to be created which we will use later. The second is the name of the trace values, taken from the log header, of the root height of the tallest gene. Now run BEAST again:

* If generation time was 10 years, 1M years would be 100,000 generations, and so 1 Ne would be 100,000 individuals.

```
beast -overwrite alld.xml
```

After completion we can generate two more plots. For the first use the following options:

```
../scripts/popGraphFromCSV.py --logy --hist ./mystery.csv ./mystery-hist.png
```

The result is figure 15. It includes a histogram describing the locations of the demographic

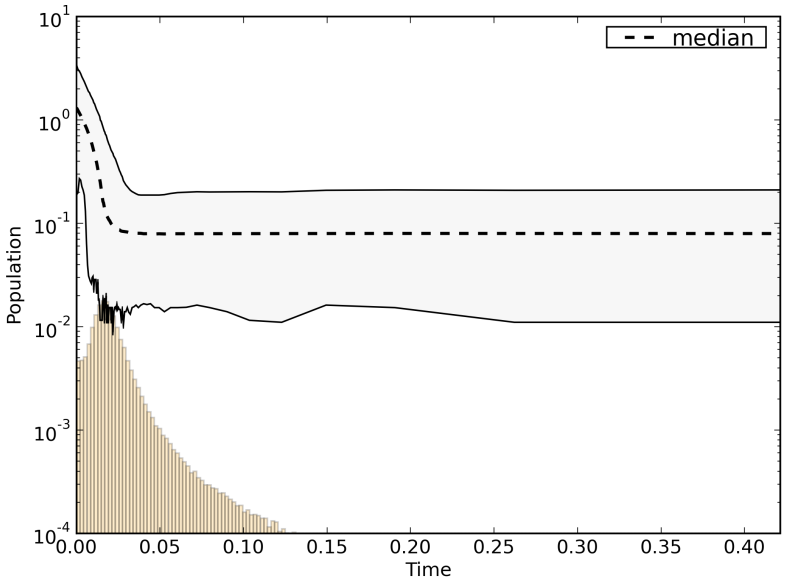


Figure 15: An EBSP plot with a histogram of the location of the last X axis point.

functions X-axis points. The EBSP demographic function is piecewise (either linear or constant), that is constructed by "joining dots". The X-axis values of the dots are not evenly spaced, and this histogram shows their locations. The height of each bar is proportional to the number of demographic functions which had an X point at that interval of time. As you can see the plot X range, which is based on the mean of the gene trees root time, extends a fair bit beyond that. In fact, the values beyond 0.12-0.13 are based on one or two data points, and the long constant tail is an artifact of the prior, not the data. One possible explanation is that those mammals experienced a bottleneck at this point, perhaps severe enough so that no amount of genes will let use see beyond this point.

For the other figure we will use the generated file '`mystery.alld.txt`'.

```
../scripts/popGraphFromCSV.py --logy --alpha 0.05 --alldem mystery.alld.txt \
                              ./mystery.csv ./mystery-05.png
```

The result is in figure 16. This gives us a view of the full posterior – all the samples that are summarized by the median and credible intervals lines. Here it does not add much beyond the median and credible summary, but look at figure 17 from another data set. Here we can see that while the upper 95% credible intervals are high, the signal is in fact stronger than what you may expect from the 95% intervals alone.

And last, there are various options to make the plot come out better in a publication. You can control the figure size, fonts, font sizes and figure ratio.

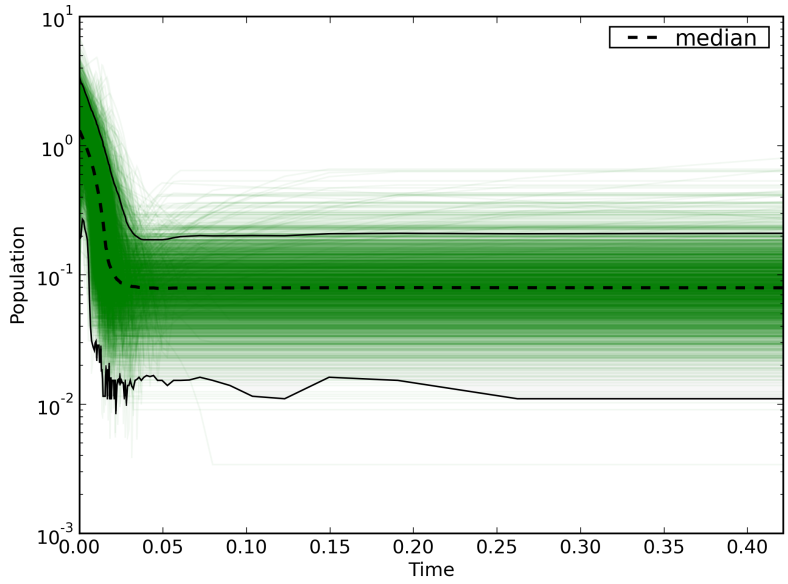


Figure 16: An EBSP plot of all demographics.

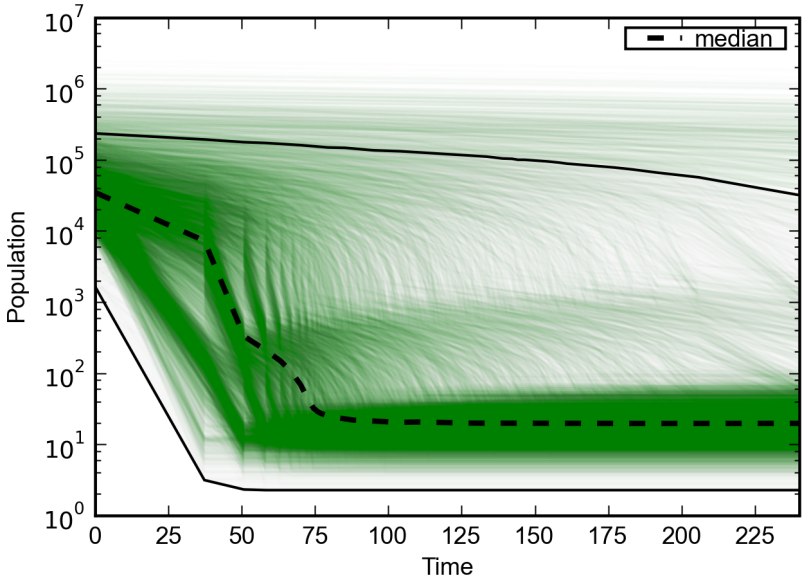


Figure 17:

```
../scripts/popGraphFromCSV.py --logy --width 12cm --ratio .85 --font Arial \
                      --fontsize 10 --ticks 10 ./mystery.csv ./mystery1.png
```

Figure 18 shows the plot. It may take some tweaking to get an acceptable result. You can probably improve the settings above as well.

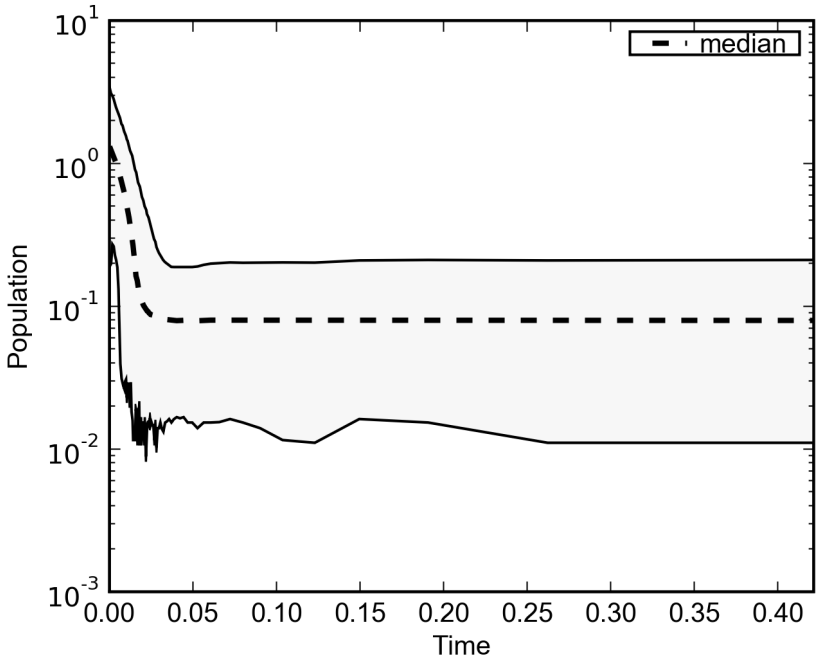


Figure 18:

References

[1] Alexei J Drummond and Andrew Rambaut. Beast: Bayesian evolutionary analysis by sampling trees. *BMC Evol Biol*, 7:214, 2007.

[2] J. Heled and A.J. Drummond. Bayesian inference of population size history from multiple loci. *BMC Evolutionary Biology*, 8(1):289, 2008.

[3] B. Nabholz, S. Glemin, and N. Galtier. Strong variations of mitochondrial mutation rate across mammals–the longevity hypothesis. *Molecular biology and evolution*, 25(1):120, 2008.