A Rough Guide to BEAST 1.4

Alexei J. Drummond[1], Simon Y.W. Ho,
Nic Rawlence and Andrew Rambaut[2]

[1]Department of Computer Science
The University of Auckland, Private Bag 92019
Auckland, New Zealand
alexei@cs.auckland.ac.nz

[2]Institute of Evolutionary Biology
University of Edinburgh
Edinburgh, United Kingdom
a.rambaut@ed.ac.uk

July 6, 2007

Contents

1 Introduction

BEAST is a cross-platform program for Bayesian MCMC analysis of molecular sequences. It is orientated towards rooted, time-measured phylogenies inferred using strict or relaxed molecular clock models. It is intended both as a method of reconstructing phylogenies and as a framework for testing evolutionary hypotheses without conditioning on a single tree topology. BEAST uses MCMC to average over tree space, so that each tree is weighted proportional to its posterior probability. We include a simple to use user-interface program for setting up standard analyses and a suite of programs for analysing the results. There are three main areas of research for which the BEAUti/BEAST package is particularly applicable. These areas are species phylogenies for molecular dating, coalescent-based population genetics and measurably evolving populations (ancient DNA or time-stamped viral sequence data sets).

2 BEAUti

BEAUti (Bayesian Evolutionary Analysis Utility) is a graphical software package that allows the creation of BEAST XML input files. The exact instructions for running BEAUti differ depending on which computer system you are operating. Please see the README text file that was distributed with the version you downloaded. Once running, BEAUti will look similar irrespective of which computer system it is running on.

2.1 Importing the NEXUS input file

In the top left hand corner of the BEAUti window is the "File" menu. From the "File" menu select "Import NEXUS". A window will appear, allowing you to select your NEXUS input file.

2.2 Data panel

Once your NEXUS input file has been imported into BEAUti, the "Data" window will appear with your sequence information displayed.

2.2.1 Name column

This column contains a unique name for each DNA sequence.

2.2.2 Date column and time-stamped data

This section is only important for researchers interested in ancient DNA (aDNA) or time-stamped data sets (generally virus sequence data). The "Date" refers to either the date or the age of the sequences. The default date for all taxa is assumed to be zero. This will be correct if all your sequences were sampled at approximately the same time point. For aDNA "Date" will typically be in radiocarbon years, though for the purposes of analysis, "Date" in years will suffice. For radiocarbon dates, only enter the absolute date value, not the associated error. The dates entered need to be specified as "Years" and "Before the present" from the "Dates specified as" menu. For viral data sets, dates

will most commonly be calendar years (e.g., 1984, 1989, 2007, etc.) or perhaps months or even days since the start of the study and need to be specified as "Years", "Months" or "Days" and "Since some time in the past". Although BEAUTi allows you to specify the units for these dates, this is simply to make a record of the units for reference. Whatever units of time you use to specify the dates will then be used throughout the analysis. For example, if you give your dates in millions of years (e.g., 0.1 representing 100,000 years) then all the reported dates and rates will be given as My (e.g., a reported rate of 0.01 will have units of substitutions per My and thus be 1.0E-8 substitutions per year).

The "Clear Dates" button resets all dates in the Date column to zero.

Sometimes it is more convenient to use the "Guess Dates" option rather than enter the date values manually into the "Dates" column. This option guesses the sequence dates from the numerical information contained within the taxon name. If the taxon name contains more than one numerical field then you can specify BEAUti to find the field that corresponds to the sampling date by specifying the order that the date field comes (first, last, etc.) or specifying a prefix (a character that comes immediately before the date field in each name). You can also add a fixed value to each guessed date.

2.2.3 Ancient DNA and radiocarbon dates

Most aDNA data sets will be a combination of dated and undated sequences. This poses a problem when analysing these data sets with BEAST. There are a couple of options available to resolve this. It is important however not to leave a date for any aDNA sequence as zero. BEAST will assume that this sequence has an age of zero (i.e., is a modern sequence), which will bias the parameter estimates such as mutation rate, divergence times and population sizes. There are a few of options:

1. Exclude non-dated aDNA sequences from the analysis. Sequences can be highlighted in the "Data" window and then deleted by selecting "Delete" from the "Edit" menu.

2. Have two to three carbon dated aDNA sequences from each sub fossil deposit, or find published dates for the sub fossil deposit. Use these dates to calculate an average age for the deposit.

3. If the first two options are not available, the nature of the deposit and the local geography can be used to calculate an average age of the deposit/aDNA sequences. For example: In New Zealand, sea levels after the last glaciation did not stabilize at present levels until 6,000 years ago. Once stabilization occurred (and not before), coastal sand dunes formed, with swamps forming behind these dunes, trapping moa. Thus the majority of coastal swamp deposits are less than 6,000 years old, with an average age of around 3,000 years. Another example is: During the Holocene species "A" lived in the geographical area "A". However, from published data we know that during the height of the last glacial cycle of the Pleistocene ice age (approximately 20,000-10,000 years ago) species "A" also lived in the geographical area "B". Thus any undated sub fossil bones of species "A" from geographical area "B" have to have an average age of 15,000 years. Finally, glacial U-shaped valleys are an indication that the valley

was under a glacier during the height of the last glacial cycle. These environments did not open up for colonization by plants and animals until 10,000-14,000 years ago. Thus any sub fossil remains found in these valleys have to be younger than 10,000-14,000 years with an average age of 5,000-7,000 years.

2.2.4 Height column

This refers to the height (age) of each sequence relative to the youngest sequence. For non-dated sequences, all heights will be zero. For time-stamped data BEAUti will designate the youngest (most modern) sequence as height zero and calculate the age/height of all other dated sequences relative to this. This information, along with the mutation rate will be used to estimate the age/height of internal nodes in the tree such as the `treeModel.rootHeight` (the age of the root of the tree).

For modern sequences, dates of divergence are set by creating priors on either internal nodes or the overall rate of substitution.

2.2.5 Sequence column

This column shows the DNA sequence alignment specified in the NEXUS input file.

2.2.6 Translation options

This is relevant if your analysis concerns protein coding genes and you want to perform an analysis based on amino acid sequences. If you are analysing non-coding sequence data such as mitochondrial DNA control region, then leave this option set to "None". This combo box specifies the amino acid translation code for the DNA sequence data. The options refer to what organism (e.g., "Vertebrate", "Yeast" or "Bacterial") and whether the sequence data is nuclear (for vertebrates select "Universal") versus mitochondrial (for vertebrates select "Vertebrate Mitochondrial"). Selecting these options will translate the DNA sequence into the amino acid sequence using the specified translation code. If your sequences are amino acid sequences, then this option will be disabled.

2.3 Taxa panel

This window allows you to set up taxon subsets within the sequence data. By setting up these subsets you define a set of taxa. This will also allow you to log the time to most recent common ancestor t_{MRCA} for each taxon subset and also set prior distributions on the corresponding divergence times. The resulting tMRCAs in the log file will be specified in the same units as those specified for your DNA sequences (i.e., radiocarbon years for aDNA data sets or for non-dated data sets whatever time units the rate/date priors are specified in). These taxon subsets can represent different species in multi-species analyses or perhaps geographically isolated populations within a species. It is important to note that setting up a taxon subset does not guarantee that this group will be monophyletic with respect to other taxa in the resulting MCMC analysis. Therefore the corresponding MRCA may in fact contain other descendants than just those specified in the taxon set.

You can set up taxon subsets as follows. In the bottom left hand corner of the screen are a "+" and "-" button. Selecting the "+" button will import your named sequences into the "Excluded Taxa" column. By selecting specific sequences, then clicking the right-hand facing arrow, these sequences will be imported into the "Included Taxa" column and vice versa. The "untitled" taxon label in the "Taxon Sets" column must be labelled with a specific name to designate the new taxon subset. Taxon sets can be added by clicking the "+" button and removed by clicking the "-" button. Taxa in the "Included Taxa" group will be used to date the t_{MRCA}, while taxa in the "Excluded Taxa" column may or may not be in the same clade.

2.4 Model panel

This window allows you to specify DNA or amino acid substitution models, partition protein coding sequence into codon positions, unlink the substitution model and rate heterogeneity across codon positions (all parameters can be either shared or made independent among partitions in the sequence data), fix the substitution rate and finally select the molecular clock model. Depending on whether your data is nucleotide or amino acid sequence (or nucleotides translated into amino acids) the options will differ. The substitution models used in BEAST will be familiar to users of other Bayesian and likelihood-based phylogenetics software.

2.4.1 Substitution model

Substitution models describe the process of one nucleotide or amino acid being substituted for another. There are two DNA substitution models available in BEAUti: the Hasegawa-Kishino-Yano (HKY) model and the General Time Reversible (GTR) model. Other substitution models can be achieved by editing the BEAST XML file generated by BEAUti. For nucleotide data, all the models that are nested within the GTR model (including the well known HKY85 model) can be specified by manually editing the XML. When analysing protein coding data the Goldman and Yang model can be used to model codon evolution [24]. For the analysis of amino acid data the following replacement models can be used: Blosum62, CPREV, Dayhoff, JTT, MTREV and WAG (ref).

2.4.2 Site heterogeneity model

This allows the refinement of the HKY or GTR model to allow different sites in the alignment to evolve at different rates. The "None", "Gamma", "Invariant Sites" and "Gamma + Invariant Sites" options in this menu help explain among site rate heterogeneity within your data.

Selecting "None" specifies a model in which all sites are assumed to evolve at the same rate. For most data sets, this will not be the case, however for some alignments there is very little variation and the equal rates across sites model can't be rejected.

Selecting "Gamma" will permit substitution rate variation among sites within your data (i.e., the substitution rate is allowed to vary so that some sites evolve slowly and some quickly). The shape parameter "alpha" of the Gamma distribution specifies the range of the rate variation among sites. Small alpha values

(< 1) result in L shaped distributions, indicating that your data has extreme rate variation such that most sites are invariable but a few sites have high substitution rates. High alpha values result in a bell shaped curve, indicating that there is little rate variation from site to site in your sequence alignment. When alpha reaches infinity, all sites have the same substitution rate (i.e., equivalent to "None"). If the analysis concerns protein coding DNA sequences, the estimated gamma distribution will generally be L-shaped. If the codons are however partitioned into 1st, 2nd and 3rd positions, 1st and 2nd will generally have a lower alpha value than the 3rd.

Selecting "Invariant Sites" specifies a model in which some sites in your data never undergo any evolutionary change while the rest evolve at the same rate. The parameter introduced by this option is the proportion of invariant sites within your data. The starting value of this parameter must be less than 1.0, or BEAST will fail to run.

Finally, selecting "Gamma and Invariant Sites" will combine the two simpler models of among-site rate heterogeneity so that there will be a proportion of invariant sites and the rates of the remaining sites are assumed to be Γ-distributed.

2.4.3 Number of Gamma categories

This combo box allows the user to choose the number of categories for the discrete approximation of the Gamma distribution [25]

2.4.4 Partitioning into codon positions

BEAST provides the ability to analyse multiple data partitions simultaneously which share or have separate parameters for each partition. If the analysis concerns just non-coding DNA like mtDNA control region, select "Off" from the menu. Partitioning is useful when combining multiple genes (e.g., cyt b and COI), protein and non-coding sequence data (control region and cytochrome b), and nuclear and mitochondrial data, or to allocate different evolutionary processes to different regions of a sequence alignment like codon positions. By partitioning your data, this allows more information from the data set to be extracted. In BEAUti, you can only partition into 1st, 2nd and 3rd codon positions. All other partitioning must be done by editing the XML file.

There are two choices. Partitioning into "(1+2)+3" keeps the 1st and 2nd positions in one partition (slower substitution rate due to their constrained nature in coding for amino acids) and the 3rd position in a separate partition (faster substitution rate due to the increased redundancy in the genetic code of the 3rd codon position). Partitioning into "1+2+3" allows each codon position to have its own substitution rate. This assumes that the data is aligned on codon boundaries, so that every third site in the alignment is the third position in a codon for all sequences in the alignment.

Unlinking substitution model across codon positions will instruct BEAST to estimate a separate transition-transversion ratio (kappa parameter) for HKY or separate relative rate parameters for GTR for each codon position. Unlinking rate heterogeneity model will instruct BEAST to estimate the among-site rate heterogeneity parameters independently for each codon position.

2.4.5 Use SRD06 Model

When this button is pressed BEAUti selects a particular combination of the above settings which represent the model suggested by Shapiro et al [**?**]. This model links 1st and 2nd codon positions but allows the 3rd positions to have a different relative rate of substitution, transition-transversion ratio and gamma-distributed rate heterogeneity. This model has fewer parameters than GTR + gamma + invariant sites but has been found to provide a better fit for protein-coding nucleotide data.

2.4.6 Fix mean substitution rate

This option is relevant when you have no fossil calibration data and want to calibrate the data set using a known substitution/mutation rate. This will in effect calibrate the phylogeny with an external rate and will mean abandoning any errors associated with this rate. If you want to calibrate your phylogeny with a rate estimate that includes uncertainty then you should unselect this option and provide a prior distribution for the rate in the Priors panel (see section 2.5).

Setting this to 1.0 will result in the ages of internal nodes being estimated in units of substitution/site, which is often appropriate when the objective of the analysis is phylogenetic reconstruction and the time frame of the phylogeny is not of interest.

2.4.7 Molecular clock rate variation model

This allows you to select the appropriate model for rate variation among branches in the tree. The model you select will be used to estimate the substitution rate for each node of the tree, the t_{MRCA} of taxon groups, and the `treeModel.rootHeight` parameter (which represents the t_{MRCA} for the root of the phylogeny). There are currently three options in BEAUti: "Strict Clock", "Relaxed Clock: Uncorrelated Exponential" and "Relaxed Clock: Uncorrelated Lognormal". A strict clock assumes a global clock rate with no variation among lineages in a tree. However, biology is generally not that simple. Often the data will best fit a relaxed molecular clock model. Relaxed molecular clock models (There are other models - one under development that will be included in later versions of BEAUti will be the Random Local Molecular Clock model) assume independent rates on different branches, with one or two parameters that define the distribution of rates across branches. The relaxed molecular clock models in BEAST are called "uncorrelated" because there is no *a priori* correlation between a lineage's rate and that of its ancestor [31].

The strict molecular clock is the basic model for rates among branches supported by BEAST. Under this model the tree is calibrated by either:

1. Specifying a substitution rate (this can be done either by fixing the mean substitution rate to a designated value or by using a prior on the `clock.rate` parameter in the "Priors" panel) or

2. Calibrating the dates of one or more internal nodes (by specifying a prior on the t_{MCRA} of a taxon subset or the `treeModel.rootHeight`). This allows the divergence dates of clades (defined either as a monophyletic

grouping or as the t_{MRCA} of a specified taxon subset) to be calculated based on the best fit of a single mutation rate across the whole tree.

When using the relaxed molecular clock models, the rate for each branch is drawn from an underlying exponential or lognormal distribution. We recommend the use of the uncorrelated relaxed lognormal clock as this gives an indication of how clock-like your data is (measured by the `ucld.stdev` parameter). If the `ucld.stdev` parameter estimate is close to 0.0 then the data is quite clock-like. If the `ucld.stdev` has an estimated value much greater than 1.0 then your data exhibits very substantial rate heterogeneity among lineages. This pattern will also be true for the coefficient of variation parameter.

Note: To test MCMC chain performance in the first run of BEAST on a new data set, it is often a good idea to start with a relatively simple model. In the context of divergence dating this might mean running a strict molecular clock (with informative priors on either `clock.rate`, one or more t_{MRCA}s, or `treeModel.rootHeight`). If BEAST can't produce an adequate sample of the posterior under a simple model, then it is unlikely to perform well on more complicated substitution and molecular clock models.

2.5 Priors panel

The Priors panel allows the user to specify informative priors for all the parameters in the model. This is both an advantage and a burden. It is an advantage because relevant knowledge such as fossil calibration points within a phylogeny can be incorporated into the analysis. It is a burden because when no obvious prior distribution for a parameter exists, it is your responsibility to ensure that the prior selected is not inadvertently influencing the posterior distribution of the parameter of interest.

2.5.1 Tree priors

When sequences have been collected from a panmictic intraspecific population there are various coalescent tree priors that can be used to model population size changes through time. Under the coalescent assumption BEAST allows a number of parametric demographic functions of population size through time: constant size, exponential growth, logistic growth and expansion growth. Which one you choose depends on the population you are analysing and the demographic assumptions you wish to make.

In addition, the Bayesian skyline plot (BSP) [29] is available, which calculates the effective breeding population size (N_e) through time (up to a constant related to the generation length in the time units of the analysis). However, the BSP should only be used if the data are strongly informative about population history, or when the demographic history is not the primary object of interest and a flexible coalescent tree prior with minimal assumptions is desirable. This coalescent-based tree prior only requires you to specify how many discrete changes in the population history are allowed. It will then estimate a demographic function that has the specified number of steps integrated over all possible times of the change-points and population sizes within each step to calculate a function of N_e through time [29]. Two variants of the BSP are provided, the "Stepwise" model in which the population is constant between

change-points and then jumps instantaneously and the "Linear" model in which the population grows or declines linearly between change-points.

All the demographic models listed above are parametric priors on the ages of nodes in the tree, in which the hyperparameters (e.g., population size and growth rate in the case of the exponential growth model) can be sampled and estimated.

For species-level phylogenies, coalescent priors are generally inappropriate. In this case, we suggest that you use the Yule tree prior. The Yule tree prior assumes a constant speciation rate per lineage. This prior has a single parameter (`yule.birthRate`) that represents the average net rate of lineage birth. Under this prior branch lengths are expected to be exponentially distributed with a mean of $\texttt{yule.birthRate}^{-1}$.

2.5.2 UPGMA starting tree

The UPGMA tree is a useful option when you have lots of constraints and priors on your starting tree that must be satisfied.

2.5.3 Parameters

Crucial to the interpretation of all BEAST parameters is an understanding of the units that the tree is measured in. The simplest situation occurs when no calibration information is available, either from knowledge of the rate of evolution of the gene region, or from knowledge of the age of any of the nodes in the tree. If this is the case the rate of evolution is set to 1.0 and the branch lengths in the tree are then in substitutions per site. However if the rate of evolution is known in substitutions per site per unit time, then the genealogy will be expressed in the relevant time units. Likewise, if the age of one or more nodes (internal or external) are known then this will also provide the units for the rest of the branch lengths and the rate of evolution. Note that only the set of parameters that are currently being used (as defined by the model settings) will be shown in the table. For example, if the rate of substitution is fixed (in the "Model" section) then the `clock.rate` parameter (or the `ucld.mean` if the relaxed clock is selected) will not be available. With this in mind, the following table lists some of the parameters that can be generated by BEAUti, their interpretation and units.

`clock.rate` — The rate of the strict molecular clock. This parameter only appears when you have selected the strict molecular clock in the model panel. The units of this parameter are in substitutions per site per unit time. If this parameter is fixed to 1.0 (using the *Fix mean substitution rate* option in the Model panel) then the branch lengths in the tree will be in units of substitutions per site. However, if, for example, the tree is being calibrated by using fossil calibrations on internal nodes and those fossil dates are expressed in millions of years ago (Mya), then the `clock.rate` parameter will be an estimate of the evolutionary rate in units of substitutions per site per million years (Myr).

`constant.popSize` This is the coalescent parameter under the assumption of a constant population size. This parameter only appears if you select a constant size coalescent tree prior. This parameter represents the product of effective population size (N_e) and the generation length in units of time (τ). If time is measured in generations this parameter a direct estimate of N_e. Otherwise it is a composite parameter and an estimate of N_e can be computed from this parameter by dividing it by the generation length in the units of time that your calibrations (or `clock.rate`) are defined in. Finally, if `clock.rate` is set to 1.0 then `constant.popSize` is an estimate of $N_e\mu$ for haploid data such as mitochondrial sequences and $2N_e\mu$ for diploid data, where μ is the substitution rate per site per generation.

`covariance` If this value is significantly positive, then it means that within your phylogeny, branches with fast rates are followed by branches with fast rates. This statistic measures the covariance between parent and child branch rates in your tree in a relaxed molecular clock analysis. If this value spans zero, then branches with fast rates and slow rates are next to each other. It also means that there is no strong evidence of autocorrelation of rates in the phylogeny.

`exponential.growthRate` This is the coalescent parameter representing the rate of growth of the population assuming exponential growth. The population size at time t is determined by $N(t) = N_e \exp(-gt)$ where t is in the same units as the branch lengths and g is the `exponential.growthRate` parameter. This parameter only appears if you have selected a exponential growth coalescent tree prior.

`exponential.popSize` This is the parameter representing the modern day population size assuming exponential growth. Like `constant.popSize`, it is a composite parameter unless the time scale of the genealogy is in generations. This parameter only appears if you have selected a exponential growth coalescent tree prior.

`gtr.{ac,ag,at,cg,gt}` These five parameters are the relative rates of substitutions for $A{\leftrightarrow}C$, $A{\leftrightarrow}G$, $A{\leftrightarrow}T$, $C{\leftrightarrow}G$ and $G{\leftrightarrow}T$ in the general time-reversible model of nucleotide substitution [22]. In the default set up these parameters are relative to $r_{C\leftrightarrow T} = 1.0$. These parameters only appear if you have selected the GTR substitution model.

`hky.kappa` This parameter is the transition/transversion ratio (κ) parameter of the HKY85 model of nucleotide substitution [23]. This parameter only appears if you have selected the HKY substitution model.

`meanRate` This statistic is logged when a relaxed molecular clock is use and it is the estimated number of substitutions per site across the whole tree divided by the estimated length of the whole tree in time. It has the same units as `clock.rate` parameter. If r_i is the rate on the ith branch and t_i is the length of time in calendar units for the ith branch then $b_i = r_i t_i$ is the branch length in substitutions per site and the `meanRate` is calculated as $\sum_i b_i / \sum_i t_i$.

`siteModel.alpha` This parameter is the shape (α) parameter of the Γ distribution of rate heterogeneity among sites [25]. This parameter only appears when you have selected Gamma or Gamma+Invariant Sites in the site heterogeneity model.

`siteModel.pInv` This parameter is the proportion of invariant sites (p_{inv}) and has a range between 0 and 1. This parameter only appears when you have selected "Invariant sites" or "Gamma+Invariant Sites" in the site heterogeneity model. The starting value must be less than 1.0.

`treeModel.rootHeight` This parameter represents the total height of the tree (often known as the t_{MRCA}). The units of this variable are the same as the units for the branch lengths in the tree and will depend on the calibration information for the rate and/or dates of calibrated nodes.

`ucld.mean` This is the *mean of the branch rates* under the uncorrelated lognormal relaxed molecular clock and is similar but not the same as the mean number of substitutions per site per unit time (see `meanRate` above). If r_i is the rate on the ith branch then `ucld.mean` is $\frac{1}{2n-2}\sum_{i=1}^{2n-2} r_i$ and does not take into account the fact that some branches are longer than others. This parameter can be in real space or in log space depending on the BEAST XML. However,under default BEAUti options for the uncorrelated log-normal relaxed clock this parameter has the same units as `clock.rate`. If you want to constrain the mean rate of the relaxed clock with a prior you should either set a prior on `ucld.mean` or `meanRate` but not both (as they are very highly correlated via the r_i parameters.

`ucld.stdev` This is the standard deviation (σ) of the uncorrelated lognormal relaxed clock (in log-space). If this parameter is 0 there is no variation in rates among branches. If this parameter is greater than 1 then the standard deviation in branch rates is greater than the mean rate. This is also the case for the coefficient of variation. When viewed in Tracer, if the coefficient of

variation frequency histogram is abutting against zero, then your data can't reject a strict molecular clock. If the frequency histogram is not abutting against zero then there is among branch rate heterogeneity within your data, and we recommend the use of a relaxed molecular clock.

`yule.birthRate` This parameter is the rate of lineage birth in the Yule model of speciation. If `clock.rate` is 1.0 then this parameter estimates the number of lineages born from a parent lineage per substitution per site. If the tree is instead measured in, for example, years, then this parameter would be the number of new lineages born from a single parent lineage per year.

`tmrca(`*taxon group*`)` This is the parameter for the t_{MRCA} of the specified taxon subset that you specified in the "Taxa" panel. The units of this variable are the same as the units for the branch lengths in the tree and will depend on the calibration information for the rate and/or dates of calibrated nodes. There will a `tmrca(`*taxon group*`)` parameter for each taxon subset specified. Setting priors on these parameters and/or `treeModel.rootHeight` parameter will act as calibration information.

2.5.4 Priors

Under the "priors" column is listed the type of prior that is being utilized for each parameter in the model. Choosing the appropriate prior is important. For example, if your sequences are all collected from a single time point, then the overall evolutionary rate must be specified with a strong prior. The units of the prior will then determine the units of the node heights in the tree, the age of the MRCA and the units of the demographic parameters such as population size and growth rate. Depending on the type of prior that you want, there are several choices. Clicking on the prior for the parameter in question brings up a menu with the following options. For divergence time estimation, calibrations are made by placing priors on the `treeModel.rootHeight` and `tmrca(`*taxon group*`)` parameters. There are a number of prior distributions that may be appropriate:

Uniform This prior allows you to set up an upper and lower bound on the parameter. For example you could set an upper and lower bound for the constant.popSize parameter. The initial value must lie between the upper and lower bounds.

Normal This prior allows the parameter to select values from a specified normal distribution, with a specified mean and standard deviation.

Lognormal This prior allows the parameter to select values from a specified lognormal distribution, with a specified mean and standard deviation (in log units). In addition it is possible to specify a translated lognormal distribution, so that the whole distribution can have a

lower limit other than 0.0. For example, a fossil that calibrates the node that cannot be younger than 100 Mya but can be older. By changing the values of the LogNormal Mean, LogNormal Stdev (which represent values in log space) and Zero offset you can create a distribution that matches your prior in real space (non-log space, the values at the bottom of the window). This prior can also be used on `clock.rate` and `constant.popSize` parameters. This is ideal if you have a small population that is highly genetically structured with deep divergences within the population, as this can artificially increase the N_e significantly. In addition, N_e is always an underestimate of the true population size, so a lower bound is difficult to determine. At present, this prior in BEAUti is somewhat inconvenient to use as the offset is in units whereas mean and standard deviation are in log units. This will be improved for the next version.

Tree prior If no calibration information is specified through a parametric prior, a parameter such as `treeModel.rootHeight` or `tmrca(`*taxon group*`)` will still have a prior distribution via the selected tree prior. This option signifies that fact.

In addition there are a number of other priors that can be specified including Exponential, Gamma and Jeffreys.

2.6 Operators panel

Operators act on the given parameters in the BEAST analysis, determining how the MCMC chain proposes new states to move to. Appropriate choice of weights and tuning parameter values will allow the MCMC chain to reach equilibrium/stationary phase faster and sample the target distribution more efficiently. Each parameter in the substitution model has one or more operators. A scale operator scales the parameter up or down by a random scale factor, with the tuning parameter deciding the range of scale factors to choose from. The random walk operator adds or subtracts a random amount (δ) to or from the parameter. Again the tuning parameter (window size, w) is used to specify the range of values that δ can take $\delta\ Uniform(-w/2, w)$. The uniform operator simply proposes a new value uniformly within a given range.

On the left hand side of this window is listed the parameters that are going to be operated including the phylogeny itself which has its own set of operators. Next to this is the type of operator that will be used.

2.6.1 Tuning

The tuning column gives the tuning setting to the operator. Some operators do not have a tuning setting so have a N/A. Changing the tuning setting will set how large a move that operator will make which will affect how often that change is accepted by the MCMC algorithm.

2.6.2 Weighting

The weight column specifies how often each operator is used to propose a new state in the MCMC chain. Some parameters have very little interaction with

the rest of the model and as a result tend to be estimated very efficiently. An example of such a parameter is `hky.kappa` which though efficiently estimated, requires a complete recalculation of the likelihood of the data whenever it is changed. Giving these parameters lower weights can improve the computational efficiency of the run.

The efficiency of the MCMC chain can often be improved by altering the weight of the operators that work on the `treeModel`. For example, if you are analysing x sequences, a very rough rule of thumb is that you should set the weight of the each of: `upDownOperator`, `uniformOperator` on `internalNodeHeights`, `narrowExchangeOperator`, `subtreeSlideOperator` to $x/2$. You should also set the weight of the `wilsonBaldingOperator` and the `wideExchangeOperator` to $\texttt{min}(1, x/10)$. These are rough guidelines and other weights may work better, but we have found these guidelines to work reasonably well. The authors of BEAST plan to perform a more systematic study of the performance of different combinations of proposals and weights so we can provide more guidance in this area in the future. Any assistance in this endeavor would be greatly appreciated.

2.6.3 Auto-optimize

The "auto optimize" option will automatically adjust the tuning parameters of operators as the MCMC algorithm runs, to try to achieve maximum efficiency. We recommend that you choose this option. The criterion that our auto-optimization method uses is a target acceptance probability for each operator. The target acceptance probability is 0.25 for most operators. The idea is that proposals for new parameter values should be bold enough that they explore the parameter space rapidly, without being so large that the proposals are never accepted. By tuning an operator so that it is accepted 25% of the time we find that the moves are big enough to explore the space while still allowing regular changes to the MCMC state.

At the end of the MCMC run, a report on the performance of the operators will be given in the BEAST console. This report includes the proportion of times that each operator was accepted, the final values of these tuning settings and whether they were at the right level for the analysis and suggestions for changes to these values. These operator tuning parameters can be changed in order to minimize the amount of time taken to reach optimum performance in subsequent runs.

Note: changing the tuning parameters of the operators will not change the results of the analysis - it will only affect the efficiency of the sampling of the posterior distribution. Better tuning parameters (and weights) will lead to faster convergence and better mixing of the MCMC chain, which means that the MCMC run can be run for fewer generations to achieve the same Effective Sample Size (ESS).

2.7 MCMC options

The MCMC panel allows you to set the number of generations the MCMC algorithm will run for, how often the data is logged to file and to name the output files that BEAST will store the data in.

2.7.1 Length of chain

This is the number of generations that the MCMC algorithm will run for. The length of chain depends on the size of the data set, the complexity of the model and the quality of the sample required. The aim of setting the chain length is to achieve a reasonable ESS, especially for the parameters of interest. A very very rough rule of thumb is that for x taxa/individuals you need a generation time/chain length proportional to x^2. Thus if, for 100 sequences, a 30,000,000 step chain gives good results then for 200 similar sequences we may need a chain of 120,000,000 steps.

2.7.2 Logging options

The "*Echo state to screen*" option determines how often a summary of the state is outputted to the BEAST console window (e.g., every 1,000 generations). This option is only important for people that have enough spare time to monitor BEAST's progress as it runs ;-). The "*Log parameters every*" option determines how often parameter values are written to the log file (e.g., every 100 generations). Dividing the number of generations in the MCMC chain by the value specified for "*Log parameters every*", will give you the sample size at the end of the BEAST run. Ideally, you should aim for a sample size of between 1000 and 10,000 logged parameter values. Logging more samples will simply produce very large output files which may be difficult to analyse in other programs.

2.7.3 File names

The log file name, tree file name and substitution trees file name determine where the data that BEAST creates will be saved. The log file will contain the posterior sample of the parameter values specified in the BEAST XML file. The tree file will contain a posterior sample of trees (with branch lengths in chronological units) that can be viewed in TreeView or FigTree. The `(subst)tree` file will be a tree file with the branch lengths in units of substitutions. These will be saved in the same folder that the BEAST XML file is saved under (on UNIX/Linux systems these files will be saved in the current working directory).

2.7.4 Generating the BEAST input file

Finally, once you are satisfied that you have specified everything you want in the BEAST XML file, click on the "Generate BEAST File" button in the bottom right hand corner of the BEAUti window. This will generate an XML file that can be saved in a specific folder. This is the file that will be used by BEAST to execute the MCMC analysis.

You can save a separate BEAUti file by selecting the "Save" option from the "File" menu. It will also be an XML file but will not be recognized by BEAST, and is only used so that you can re-load it in BEAUti and quickly make modifications to your analysis at a later date. It is recommended that you save the BEAUti files with the extension ".beauti" to distinguish them from the BEAST input files.

3 BEAST

3.1 Input format

One of the primary motivations for providing a highly structured XML input format is to facilitate reproducibility of complex evolutionary analyses. We strongly encourage the routine publication of XML input files as supplementary information with publication of the results of a BEAST analysis. Because of the non-trivial nature of MCMC analyses and the need to promote reproducibility, it is our view that the publication of the exact details of any Bayesian MCMC analysis should be made a pre-requisite for publication of all MCMC analysis results.

3.2 Running BEAST

When you open the BEAST software, it will ask you to select your BEAST XML input file. If the XML file is correct (i.e., no XML or BEAST errors) then BEAST will run through the various commands and statistics that you have specified in the BEAST XML file. This will be followed by the pre-burn-in phase (a defined number of MCMC generations that are discarded at the very beginning of the MCMC run). During the pre-burn-in phase the operators are not auto-optimized so as to prevent the operators optimizing incorrectly due to the very different conditions at the start of the run when the tree is still random. The * symbols should extend along the dotted line towards towards 100. Once this has occurred, BEAST will start logging to screen and the log file the posterior sample values given the model specified in the BEAST XML file.

Note: you should always carry out at least two independent BEAST runs and then combine the log output files using LogCombiner or TRACER. This will help increase the ESS of your analysis and also will allow you to determine if the two independent runs (which will typically have different random starting trees) are converging on the same distribution in the MCMC run. If they do not converge on the same distribution then one or both of the runs have failed to converge on the posterior distribution.

3.3 Errors running BEAST

There are two broad types of error messages that appear in the BEAST window: XML errors and BEAST errors.

3.3.1 XML errors

The most general error message is:

```
Parsing error - the input file is not a valid XML file
```

As this error suggests, the file you have selected is probably not even an XML file. Another possible error message is:

```
Parsing error - poorly formed XML (possibly not an XML file):
The markup in the document following the root element must be
well-formed.
```

This error message means that the input file was recognized as an XML document but it contains a syntax error that prevents the XML file from being parsed.

When more detailed XML errors appear, this means that there is something wrong with the syntax of the XML file, but it is mostly well-formed. This generally only happens if the BEAST XML file has been manually edited. The error message will usually tell you what the XML error is and what line/character the error is located. A common XML error occurs when one of the elements does not have a closing tag (see next section on editing XML). We recommend that you use an XML-aware editor that will highlight all XML errors like a spell checker. This will ensure that your XML file is well formed.

3.3.2 BEAST errors

These errors should not occur unless you have edited the XML, in which case look at the trouble shooting XML section. BEAST errors can occur even if the XML is well-formed because the XML file may still not describe a valid BEAST analysis. These errors commonly include:

Spelling mistakes in the parameters names defined in `<parameter id="parameterName">` and `<parameter idref="parameterName">` which trick BEAST into thinking that there is a new parameter that has not been previously declared.

The first time you list a parameter it must be have an `id` attribute. All subsequent times you reference the same parameter, use the attribute `idref` with the same parameter name. Another possible message is:

```
Parsing error - poorly formed BEAST file
```

This message means that the input file was recognized as an XML file but contained some XML elements that BEAST could not understand. Generally all the details are listed below this error including the line/character number and what BEAST expected to see. Another common error message is:

```
Tree likelihood is zero.
```

This error message means that after considering all the constraints built into the BEAST XML file, the likelihood of the starting tree is 0 or smaller than the smallest positive number that can be represented using double precision floating points numbers. This can happen with data sets containing a large numbers of sequences (typically > 100). It can also happen if the starting tree does not conform to some of the t_{MRCA} priors involving upper or lower bounds. There are a few ways to try to fix this problem. First, in the XML file you can manually edit the starting values for your parameters to try to achieve a better likelihood for the starting tree. First and foremost, the initial values of all parameters have to be within the upper and lower bounds. If you do not explicitly specify an initial value for a parameter using the `value` attribute, BEAST will assume the starting value is zero. This can cause a problem, for example, in the case of the `clock.rate` parameter, an initial value of zero will result in a likelihood of zero for alignments with variable sites.

If this error is occurring because the random initial tree has a likelihood value that is too close to zero then another alternative is to start with a UPGMA tree. This can be specified in BEAUti or by manually editing the XML file.

A common cause of this error is if you have specified multiple calibration bounds in the analysis or a set of monophyly constraints. Often the random (and even UPGMA) starting trees will not conform to these calibration bounds or monophyly constraints and this will cause the tree likelihood to be zero. In this situation you must specify a valid starting tree in Newick format that conforms to any monophyly or node height constraints that you have specified (see section 4.2.6).

Finally, sometimes BEAST does not proceed through the pre-burn-in period (i.e., for whatever reason BEAST freezes in the pre-burn-in period). This may happen if you only have four or fewer taxa in one of your trees.

3.4 Troubleshooting BEAST

Sometimes BEAST crashes without any indication why. In these cases you will need to run BEAST from the command-line in order to get the error message from the program so that you can report the error message to the BEAST development team.

3.4.1 Running BEAST from the command line

Open the Command Prompt (or Terminal on Mac OS X) and navigate to the directory containing the BEAST executable. In Windows type the following command:

```
java -jar lib\beast.jar
```

Alternatively, on Mac OS X, use this command:

```
java -jar BEAST\ v1.4/Contents/Resources/Java/beast.jar
```

This will run BEAST as if it was double clicked, however now if BEAST crashes the crash message (called a stack trace) will be written to the Command Prompt (Terminal). This will help in diagnosing the problem.

3.4.2 Giving BEAST more memory

Sometimes users try to use very large datasets and BEAST will run out of memory. In some instances this can be remedied by running BEAST from the command line (see section 3.4.1) and adding some command line options to direct JAVA to give BEAST more memory using the `-Xms` and `-Xmx` JAVA options. For example if you wanted to give BEAST 1024Mb of memory you would use the following command:

```
java -Xms1024m -Xmx1024m -jar lib\beast.jar
```

3.5 BEAST output and results

At the end of the run report of performance of the operators will be outputted to the BEAST console window. This table will display the operators, their acceptance probabilities, the final values of the associated tuning values, and a textual message indicating whether they were successfully tuned to the right level for the given data set (low, good, slightly high, too high, etc.). The operator

tuning parameters can be changed either directly in the XML file or in the "Operator" panel in BEAUti. This level of tuning is optional, as it will not alter the results that you will get. However it will increase the ESSs in future runs.

The main output from BEAST is a simple tab-delimited plain text file format (log file) with a row for each sample (the number of rows are dictated by the log frequency specified in the BEAST XML file). When accumulated into frequency distributions, this file provides an estimate of the marginal posterior probability distribution of each parameter. This can be done using any standard statistics package or using the specially written package, TRACER. TRACER provides a number of graphical and statistical ways of analysing the output of BEAST to check performance and accuracy. It also provides specialized functions for summarizing the posterior distribution of population size through time when a coalescent model is used (e.g., Bayesian Skyline Plot).

The phylogenetic tree of each sample state is written to a separate file in either NEWICK or NEXUS format (`tree` file or `(subst)tree` file). This can be used to investigate the posterior probability of various phylogenetic questions, such as the monophyly of a particular group of organisms, or to obtain a consensus phylogeny.

3.6 Opening the BEAST log file in TRACER

It is possible to open the log file in TRACER while BEAST is still running in order to examine how the run is proceeding. However, the TRACER statistics will not be updated within TRACER as the run proceeds. To update the results in TRACER you must re-load the log file.

4 Editing BEAST XML input files

4.1 XML

XML (eXtensible Mark-up Language) is not a file format but is a simple markup language that can be used to define a file format for a particular purpose. XML is designed to be easily read by software while still being relatively easy to be read and edited by humans. BEAST understands a particular file format that has been defined using the XML markup language.

Generally, in XML, space, tab and new-lines or carriage-returns are all treated as whitespace. Whitespace characters are used to separate words in the file but it does not matter how many or in what combination they are used. Thus parts of the file can be split onto two lines or indented in an arbitrary manner.

4.2 BEAST XML format

Below is the structure of the various sections of the XML BEAST file, an explanation of their structure and how to set up the relevant parts of the file. This is by no means an exhaustive list of what can be incorporated into the BEAST XML file but will provide a guide to setting one up, and how to partition data sets into various genes and codon positions and perform a few moderately

complicated analyses. For further information please visit the following link: `http://beast.bio.ed.ac.uk/BEAST_XML_Reference`.

XML files can be constructed in NotePad, WordPad or an XML editor. It is important however that if XML files are constructed in NotePad or WordPad, that they are saved as Plain Text files and **not** Rich Text files. The majority of the following structural elements can be specified in BEAUti. However, knowing the structure behind them is an important step towards understanding how BEAST works.

The following sections are shown in roughly the order that they appear in the BEAST XML file.

4.2.1 Some general rules of XML and the BEAST format

The BEAST XML file always starts and ends with `<beast>` and `</beast>` respectively.

The characters `<` and `>` are used to bracket the beginning of an element (e.g., `<beast>`). This beginning of an element is called an open tag. The open tag may have attributes in the form of `name="value"` pairs (e.g. `<mcmc id="mcmc" chainLength="10000000" autoOptimize="true">`). The characters `</` and `>` are used to signal the end of an element (e.g., `</beast>`). The end of an element is called a close tag and cannot have attributes. Between the open and close tags of an element are its contents. In a BEAST XML file all contents are inside the `beast` element. When an element has no contents other than attributes the characters `<` and `/>` can be used instead so that the open and close tags are merged into one (e.g., `<taxon id="Medi"/>`).

The attribute `id` is used to give an element a unique identifier. The attribute `idref` is used to refer to a previously defined element that has the corresponding `id`. An `id` can be any string of characters but it is customary style to choose meaningful parameter ids that contain information about what part of the model the parameter is associated with and what the parameter represents (e.g `treeModel.rootHeight`).

All attributes of an element must always be inside double quotes.

Comments can be inserted anyware within the XML file. These comments will be completely ignored by BEAST. A comment has a special character sequence at the beginning and end.

`<!-- This is a comment -->`.

Besides describing what different parts of the XML file are for, comments can be helpful if you want to work out what command is causing a BEAST run to stall or not work properly. Placing the comment symbols (`<!--` and `-->`) around an element will cause BEAST to ignore the element. If this is done systematically for various elements (such as the priors), then it is often possible to work out which element is causing the error and thereby fix it.

4.2.2 Taxa element/block

The taxa element defines the individuals that the DNA or amino acid sequences were isolated from. This block links the sequences with the tips of a tree or sequences in different alignments together. The taxa element `<taxa id="taxa">` is a unique identifier to reference the taxa that your sequences come from. The taxa block is where you list the names of your sequences (e.g., `<taxon`

id="Medi">), dates associated with the sequences, direction the dates are measured in and the units that the dates are in (e.g., <date value="3000.0" direction="backwards" units="years"/>). In the case of aDNA data sets the direction is generally backwards (because radiocarbon dates are generally specified as ages), whereas for viral data sets, which tend to have dates specified in calendar years (e.g., 1989, 1999, 2006, etc.), the direction is forwards. For data sets in which all of the sequences are from the same time point the date element is not necessary. This element is generated in BEAUti from information in the Data panel.

```
<taxa id="taxa">
    <taxon id="Medi_3000_50">
        <date value="3000.0" direction="backwards" units="years"/>
    </taxon>
    <taxon id="Medi_1000_50">
        <date value="1000.0" direction="backwards" units="years"/>
    </taxon>
    <taxon id="Medi_7000_50">
        <date value="7000.0" direction="backwards" units="years"/>
    </taxon>
    <taxon id="Medi_2000_50">
        <date value="2000.0" direction="backwards" units="years"/>
    </taxon>
    <taxon id="Medi_1000_50">
        <date value="1000.0" direction="backwards" units="years"/>
    </taxon>
</taxa>
```

4.2.3 Defining subsets of the taxa element/block

Multiple sets of taxa can be defined by successive taxa elements. Say we have already defined the chimpanzee and bonobo taxa but want to group them together, so that BEAST creates a t_{MRCA} statistic for this group. This is done in the following manner. Remember that since you have already defined these taxon elements previously, you only need to use the `idref` attribute. Grouping taxa into a taxa element does not constrain the taxa to be monophyletic in the BEAST analysis. Defining these taxon subsets can also be done in the "Taxa" panel within BEAUti.

```
<taxa id="Pan">
    <taxon idref="chimp"/>
    <taxon idref="bonobo"/>
</taxa>
```

4.2.4 Alignment element/block

The alignment element is used to specify the multiple sequence alignment that will be analysed. Once again you must define the alignment block with a unique id and also specify the data type (`nucleotide`, `aminoAcid`) with the `dataType` attribute. Each sequence element should reference a previously defined taxon element as well as the aligned sequence (with - designating gaps). You can

directly enter the individual nucleotide sequences from an existing alignment by using the copy/paste function. Alternatively this element will be generated from information in the "Data" panel of BEAUti.

```
<alignment id="alignment" dataType="nucleotide">
    <sequence>
        <taxon idref="Medi_3000_50"/>
        TTGGCTCA
    </sequence>
    <sequence>
        <taxon idref="Medi_1000_50"/>
        TTGGCTCA
    </sequence>
    <sequence>
        <taxon idref="Medi_7000_50"/>
        TTGGCTCA
    </sequence>
    <sequence>
        <taxon idref="Medi_2000_50"/>
        TTGGCTCA
    </sequence>
    <sequence>
        <taxon idref="Medi_1000_50"/>
        TTGGCTCA
    </sequence>
</alignment>
```

Each alignment block must also have a patterns block. Once again this has a unique id. It also specifies what region of the alignment the patterns should be calculated from with the from and to attributes. The patterns element contains a reference to the corresponding alignment element, indicating that the patterns we are referring to will be calculated from the specified alignment. Again BEAUti will generate this element automatically from information in the "Data" panel.

```
<patterns id="patterns" from="1" to="383">
    <alignment idref="alignment"/>
</patterns>
```

4.2.5 Demographic model element/block

The demographic model element lets you specify a parametric model of population size to be used as part of the coalescent tree prior. The parametric models available are constant size, exponential growth, logistic growth and expansion growth. The coalescent tree prior will be used to estimate the size of a population from a sequence alignment, assuming that the sequences have been randomly sampled from a single panmictic population. In the example below we define a constant population size demographic model, which contains a `populationSize` element, which in turn contains a parameter element. It is important to ensure a reasonable starting value for the parameter and appropriate upper and lower bounds. The demographic model element will be generated based on options in the "Model" panel within BEAUti.

```
<constantSize id="constant" units="years">
    <populationSize>
        <parameter id="constant.popSize" value="100000.0"/>
    </populationSize>
</constantSize>
```

4.2.6 Starting tree block/element

This block/element specifies the starting tree to be used in the MCMC run. Starting with a random starting tree can sometimes lead to difficulties because the randomly generated tree may not satisfy all of the constraints that have been placed on it in the form of priors on tree topology and divergence times. If there are no hard priors (i.e., uniform priors on divergence times or monophyly constraints), then a random starting tree can be generated using the coalescent tree prior. The coalescent starting tree block has a unique id. It uses the taxa and demographic model element that you have specified to construct a random starting tree. This element will be generated based on options in the "Model" panel in BEAUti.

```
<coalescentTree id="startingTree">
    <taxa idref="taxa"/>
    <constantSize idref="constant"/>
</coalescentTree>
```

A second option for the starting tree is to start with a UPGMA tree. Again this can be achieved by selecting the appropriate option in the "Model" panel in BEAUti. In the XML you can also specify the root height of the tree (and thus scale the whole UPGMA tree to a certain time scale). The units of the rate and/or date priors will determine the units of this root height. By default, the UPGMA tree is constructed using a Jukes-Cantor (ref) distance matrix constructed from the sequence alignment (specified by the patterns element). If you are making this change manually in the XML file then you will also have to change the tree model element to make sure that it references the correct starting tree. This is done by deleting the `<coalescentTree idref=''startingTree_Medi''/>` command from the tree model block and replacing it with `<upgmaTree idref="startingTree"/>`

```
<upgmaTree id="startingTree" rootHeight="25">
    <distanceMatrix correction="JC">
        <patterns>
              <alignment idref="alignment"/>
        </patterns>
    </distanceMatrix>
</upgmaTree>
```

A third option for the starting tree is to specify a user-defined tree in NEWICK format. This tree can also be scaled to a different time scale automatically by specifying a `rootHeight` attribute. Starting with a user-defined tree is often necessary if there are constraints on the tree imposed by prior distributions on divergence times or the tree topology. The NEWICK format is used in many programs including PHYLIP and PAML and is embedded within the NEXUS format used by PAUP*. Once again you must delete

`<coalescentTree idref="startingTree"/>` from the tree model element and replace it with `<newick idref="startingTree"/>`.

```
<newick id="startingTree" units="years">
  ((Mus_musculus:20,Rattus_norvegicus:20):45,
  ((((Pan_paniscus:2,Pan_troglodytes:2):
  4,Homo_sapiens:6):2,Gorilla_gorilla:8):5,
  Pongo_pygmaeus:13):52);
</newick>
```

Finally you can specify a tree using XML format. This will start with an element representing the tree that contains a node element representing the root of the tree. This element will in turn contain two or more node elements representing its descendants, which in turn contain their descendant nodes. A node that represents a sampled taxon contains no descendant nodes but a taxon element (or reference to one). This is done by manually editing the XML file:

```
<tree id="Tree2">
    <node>
        <node>
            <node><taxon idref="Brazi82"/></node>
            <node>
                <node><taxon idref="ElSal83"/></node>
                <node><taxon idref="ElSal94"/></node>
            </node>
        </node>
        <node>
            <node><taxon idref="Indon76"/></node>
            <node><taxon idref="Indon77"/></node>
        </node>
    </node>
</tree>
```

4.2.7 TreeModel block/element

This element defines the node height (divergence time) parameters. It contains a reference to the starting tree element and defines various tree parameters such as `treeModel.rootHeight` (the t_{MRCA} of the tree), and internal node heights. The units of these parameters will be determined by calibration information (i.e., generally years, months or days for serial time-stamped data sets, or substitutions/site for non-time-stamped data sets without rate/date priors). BEAUti generates this block automatically.

```
<treeModel id="treeModel">
    <coalescentTree idref="startingTree"/>
    <rootHeight>
        <parameter id="treeModel.rootHeight"/>
    </rootHeight>
    <nodeHeights internalNodes="true">
        <parameter id="treeModel.internalNodeHeights"/>
    </nodeHeights>
```

```
    <nodeHeights internalNodes="true" rootNode="true">
        <parameter id="treeModel.allInternalNodeHeights"/>
    </nodeHeights>
</treeModel>
```

4.2.8 Bayesian Skyline Plot element/block

This element/block is used when an analysis involves the Bayesian Skyline Plot tree prior. An upper limit on the population size can be specified by adding the `upper` attribute to the `skyline.popSize` parameter below. This can also be done from the "Priors" panel in BEAUti. Here is an example of the XML:

```
<generalizedSkyLineLikelihood id="skyline" linear="false">
    <populationSizes>
        <parameter id="skyline.popSize" dimension="5" value="100" lower="0.0" upper="1000.0"/>
    </populationSizes>
    <groupSizes>
        <parameter id="skyline.groupSize" dimension="5"/>
    </groupSizes>
    <populationTree>
        <treeModel idref="treeModel"/>
    </populationTree>
</generalizedSkyLineLikelihood>
```

4.2.9 The t_{MRCA} statistic element/block

This element represents the t_{MRCA} for a pre-defined taxon subset. This statistic represents the divergence time of the node representing the MRCA of the given taxa (regardless of whether the taxa are monophyletic in the tree). This statistic thus allows a particular divergence time to be logged even though tree topology may be changing. By logging this statistic you can obtain a Bayesian posterior distribution of the divergence of the MRCA of the specified taxa. It has its own unique id, and the element references the treeModel that will be used to construct a phylogeny from your data.

```
<tmrcaStatistic name="time_Pan">
    <treeModel idref="treeModel1"/>
    <mrca>
        <taxa idref="Pan"/>
    </mrca>
</tmrcaStatistic>
```

You can also create a uniform, normal or lognormal prior with appropriate parameters on a t_{MRCA} statistic and thereby use it as a calibration point. This can be achieved in the "Priors" panel in BEAUti.

4.2.10 Monophyly statistic block/element

This element represents a Boolean statistic that indicates whether a predefined taxon subset is monophyletic in the tree. This statistic can be logged in a BEAST analysis to investigate how frequently the clade is sampled during the MCMC analysis. It must be done by manual editing the XML file. In the BEAST run, the monophyly statistic will return a value of 1 if the specified taxa are monophyletic on the tree and 0 otherwise. The taxa are monophyletic

if there is a node in the tree that has as descendants all the specified taxa and no others.

```
<monophylyStatistic id="panMonophyly" name="panMonophyly">
    <mrca>
        <taxa idref="Pan"/>
    </mrca>
    <treeModel idref="treeModel1"/>
</monophylyStatistic>
```

If you want to constrain a subset of taxa to always be monophyletic in the tree, then you need to create a `booleanLikelihood` element. This element returns a likelihood of 1 (true) if all of the Boolean statistics it contains are true, otherwise it returns a likelihood of 0 (false). It acts as a multiplier for the posterior. If the proposed tree satifies all the monophyly constraints then the likelihood is multiplied by 1, otherwise it is multiplied by 0 and the proposed tree is rejected. The `booleanLikelihood` element should be added to the `priors` element in the `mcmc` element. You will also have to supply a pre-specified starting tree that obeys the monophyly constraint for the MCMC analysis to start successfully.

```
<booleanLikelihood id="boolean1">
    <monophylyStatistic idref="panMonophyly"/>
</booleanLikelihood>
```

4.2.11 Coalescent likelihood block/element

This element is used to link a demographic tree prior to a treeModel. This coalescent likelihood of the specified tree given the parameters of the demographic model, and the divergence times specified by the treeModel parameters in the `populationTree` element. BEAUti automatically generates this element when a coalescent-based tree prior is chosen.

```
<coalescentLikelihood id="coalescent">
    <model>
        <constantSize idref="constant"/>
    </model>
    <populationTree>
        <treeModel idref="treeModel"/>
    </populationTree>
</coalescentLikelihood>
```

4.2.12 Molecular clock model block/element

This element defines the molecular clock model that will be used to calculate the likelihood of the tree. For a strict molecular clock there is a single parameter (the rate of the molecular clock) and the XML element looks like this:

```
<strictClockBranchRates id="branchRates">
    <rate>
        <parameter id="clock.rate" value="1.0E-5"/>
    </rate>
</strictClockBranchRates>
```

BEAST also allows two models of relaxed molecular clock (uncorrelated exponential or uncorrelated lognormal). The following XML describes the uncorrelated lognormal relaxed clock:

```
<discretizedBranchRates id="discreteBranchRates">
    <treeModel idref="treeModel"/>
    <distribution>
        <logNormalDistributionModel id="lnd" meanInRealSpace="true">
            <mean>
                <parameter id="lndMean" value="1e-2" lower="0" upper="10"/>
            </mean>
            <stdev>
                <parameter id="lndStDev" value="1e-3" lower="0" upper="10"/>
            </stdev>
        </logNormalDistributionModel>
    </distribution>
    <rateCategories>
        <parameter id="rateCategories" dimension="12"/>
    </rateCategories>
</discretizedBranchRates>
```

For exponentially distributed rates, the XML looks like:

```
<discretizedBranchRates id="discreteBranchRates">
    <treeModel idref="treeModel"/>
    <distribution>
        <exponentialDistributionModel id="ed">
            <mean>
                <parameter id="edMean" value="1e-2" lower="0" upper="10"/>
            </mean>
        </exponentialDistributionModel>
    </distribution>
    <rateCategories>
        <parameter id="rateCategories" dimension="12"/>
    </rateCategories>
</discretizedBranchRates>
```

There are several aspects of the above elements that require some attention. Firstly, in the lognormal uncorrelated relaxed clock, the `meanInRealSpace` attribute determines whether the mean of the distribution is described in standard units (`="true"`), or log units (`="false"`). This choice will have an effect on the implicit prior distribution of this parameter.

If `meanInRealSpace="true"` then the mean of the lognormal distribution should be set to some value close to the assumed rate of evolution (that is, within an order of magnitude). For example for mitochondrial protein-coding sequences calibrated in millions of years, it might be set to 10^{-2}. Since one time unit in this example represents one million years, a rate of 10^{-2} is equivalent to 1% per million years.

The `meanInRealSpace` attribute does not apply to the uncorrelated exponential relaxed clock, in which the mean is always in real space. As for the lognormal distribution, however, it is wise to make a good guess about the initial value for the mean in order to avoid a long burn-in time.

The number of dimensions for the `rateCategories` parameter should be set to the value $2N - 2$, where N is the number of taxa in the data set. So, if

there are seven taxa, the `rateCategories` parameter would have $2 \times 7 - 2 = 12$ dimensions.

4.2.13 Substitution model element

This element defines the substitution model for your data set. A frequencies element defines the base pair frequencies with reference to the data type and the alignment. This element is automatically generated based on choices made in the "Model" panel of BEAUti. Choosing the HKY substitution will generate XML that looks like this:

```
<hkyModel id="hky">
    <frequencies>
        <frequencyModel dataType="nucleotide">
            <alignment idref="alignment"/>
            <frequencies>
                <parameter id="hky.frequencies" dimension="4"/>
            </frequencies>
        </frequencyModel>
    </frequencies>
    <kappa>
        <parameter id="hky.kappa" value="1.0" lower="0.0" upper="100.0"/>
    </kappa>
</hkyModel>
```

Selecting the GTR substitution model will generate XML that looks like this:

```
<gtrModel id="gtr">
    <frequencies>
        <frequencyModel dataType="nucleotide">
            <alignment idref="alignment"/>
            <frequencies>
                <parameter id="gtr.frequencies" dimension="4"/>
            </frequencies>
        </frequencyModel>
    </frequencies>
    <rateAC>
        <parameter id="gtr.ac" value="1" lower="0" upper="500"/>
    </rateAC>
    <rateAG>
        <parameter id="gtr.ag" value="1" lower="0" upper="500"/>
    </rateAG>
    <rateAT>
        <parameter id="gtr.at" value="1" lower="0" upper="500"/>
    </rateAT>
    <rateCG>
        <parameter id="gtr.cg" value="1" lower="0" upper="500"/>
    </rateCG>
    <rateCT>
        <parameter id="gtr.ct" value="1" lower="0" upper="500"/>
```

```
    </rateCT>
</gtrModel>
```

4.2.14 Site model element

This element defines the among-site rate heterogeneity model for your data (i.e., whether you have no rate heterogeneity among sites, gamma-distributed rate heterogeneity, a proportion of invariant sites, or both gamma-distributed rate heterogeneity and invariant sites). This element combines the basic substitution model with the parameters associated with among-site rate heterogenetiy. A siteModel XML element will be generated similar to the following based on choices in the "Model" panel in BEAUti:

```
<siteModel id="siteModel">
    <substitutionModel>
        <hkyModel idref="hky"/>
    </substitutionModel>
    <gammaShape gammaCategories="4">
        <parameter id="siteModel.alpha" value="1.0" lower="0.0" upper="100.0"/>
    </gammaShape>
    <proportionInvariant>
        <parameter id="siteModel.plnv" value="0.01" lower="0.0" upper="1.0"/>
    </proportionInvariant>
</siteModel>
```

4.2.15 Tree likelihood element

This element draws together all the components involved in the tree likelihood calculation. Once again it has a unique `treeLikelihood id`. Listed under this parameter are the already defined parameters that will be used to calculate the treelikelihood. This is automatically done in BEAUti.

```
<treeLikelihood id="treeLikelihood">
    <patterns idref="patterns"/>
    <treeModel idref="treeModel"/>
    <siteModel idref="siteModel"/>
    <strictClockBranchRates idref="branchRates"/>
</treeLikelihood>
```

4.2.16 Partitioning Data

BEAUti provides an easy way to partition your alignment into codon positions (1st, 2nd and 3rd). However by editing the XML it is also possible to partition your data into different genes, nuclear versus mitochondrial DNA, coding versus non-coding or even different non-coding regions (HVR1 and HVR2 of the mtDNA control region). In BEAUti you can partition into codon positions and unlink substitution model and among-site heterogeneity parameters. Unfortunately, all other data partitioning must be done by manually editing the XML file.

First of all for multi-locus data you need each locus to have a separate alignment block. Second, you need to duplicate the patterns element, so that each partition has a patterns element. For partitioning between codon positions

the `from` attribute in a patterns element represents the codon position (1, 2 or 3) and the `every` attribute should be set to "3". Below is an example of a multi locus data set (2 genes/alignments, each split into 3 codon positions for a total of 6 partitions):

```
<patterns id="patterns1_E1" from="1" every="3">
    <alignment idref="alignment1"/>
</patterns>

<patterns id="patterns2_E1" from="2" every="3">
    <alignment idref="alignment1"/>
</patterns>

<patterns id="patterns3_E1" from="3" every="3">
    <alignment idref="alignment1"/>
</patterns>

<patterns id="patterns1_E2" from="1" every="3">
    <alignment idref="alignment2"/>
</patterns>

<patterns id="patterns2_E2" from="2" every="3">
    <alignment idref="alignment2"/>
</patterns>

<patterns id="patterns3_E2" from="3" every="3">
    <alignment idref="alignment2"/>
</patterns>
```

Finally, in order that each partition (e.g., gene, codon position, etc.) can have independent parameter estimates, the relevant elements have to be duplicated. For example, by duplicating the starting tree and `treeModel` elements, you can define a model in which each partition has an independent tree. You will also have to duplicate the `treeLikelihood` element for each partition. If you want to assume a different demographic model for each partition you will also have to duplicate the demographic model and coalescent likelihood elements. For each duplicated element you will need to create a unique id. Each partition can have as many or few independent parameters as you like. If the partitions share a common parameter, you must define the parameter in the first element that uses it and then use the `idref` attribute to refer to the parameter in subsequent elements.

4.2.17 Operators element

The operators element includes all the different types of proposals/moves that will be made during the MCMC analysis. Failing to specify any operators for a specific parameter will mean that the parameter will be fixed to its initial value, because no new values will be proposed. This is one way to fix parameters that you dont want to estimate (e.g., remove all the operators that act on the tree if you want to fix the tree topology to the starting tree). Below is an example of the contents of the `operators` element. Scale operators, swap operators,

up-down operators and uniform operators all act on the specific parameters that you have specified in the XML file. Subtree slide, narrow exchange and wide exchange operators all act on the tree. Each operator has a weight which determines how often the operator acts on the specified parameter. Most of this does not need to be changed and automatically generated based on what you have specified in BEAUti. However, if you are partitioning data then this becomes more complicated as each partition in the data needs its own set of operators.

```
<operators id="operators">
    <scaleOperator scaleFactor="0.25" weight="1" adapt="false">
        <parameter idref="gtr1.ac"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.4305" weight="1" adapt="false">
        <parameter idref="gtr1.ag"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.1853" weight="1" adapt="false">
        <parameter idref="gtr1.at"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.1853" weight="1" adapt="false">
        <parameter idref="gtr1.cg"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.1853" weight="1" adapt="false">
        <parameter idref="gtr1.gt"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.5" weight="1">
        <parameter idref="siteModel.alpha"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.5" weight="1">
        <parameter idref="siteModel.pInv"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.5" weight="1">
        <parameter idref="yule.birthRate"/>
    </scaleOperator>
    <scaleOperator scaleFactor="0.9" adapt="false" weight="1">
        <parameter idref="lndMean"/>
    </scaleOperator>
    <swapOperator autoOptimize="false" weight="5" size="1">
        <parameter idref="rateCategories"/>
    </swapOperator>
    <upDownOperator weight="4" scaleFactor="0.9">
        <up>
            <parameter idref="lndMean"/>
        </up>
        <down>
            <parameter idref="treeModel.allInternalNodeHeights"/>
        </down>
    </upDownOperator>
    <scaleOperator scaleFactor="0.5" weight="1">
        <parameter idref="lndStDev"/>
```

```
    </scaleOperator>
    <scaleOperator scaleFactor="0.5" weight="1">
        <parameter idref="treeModel.rootHeight"/>
    </scaleOperator>
    <uniformOperator weight="4">
        <parameter idref="treeModel.internalNodeHeights"/>
    </uniformOperator>
    <subtreeSlide weight="5" gaussian="true" size="1.0">
        <treeModel idref="treeModel"/>
    </subtreeSlide>
    <narrowExchange weight="1">
        <treeModel idref="treeModel"/>
    </narrowExchange>
    <wideExchange weight="1">
        <treeModel idref="treeModel"/>
    </wideExchange>
</operators>
```

4.2.18 MCMC element

This element specifies how the MCMC run will run and the output that will be produced from the run. You can specify what parameter values you want to output to file and can also put certain priors on parameters in this element within the `prior` element. The first part of this element is concerned with calculating the prior and posterior probabilities. The next section is concerned with what parameter values are logged to screen. The final section is concerned with what parameters are logged to file.

To specify a prior on a parameter you list the type of prior that you want with the specific associated values (see `logNormalPrior` below) and within the element you must refer to the parameter that you want the prior to act on. This is also where you would add the `booleanLikelihood` prior for constraining a taxon subset to be monophyletic. Below is an example of the prior element of the MCMC block. This example defines prior distributions for the `clock.rate`, `treeModel.rootHeight` and `siteModel.alpha` parameters, as well as coalescent-based tree priors for three different loci:

```
<mcmc id="mcmc" chainLength="10000000" preBurnin="30000" autoOptimize="true">
    <posterior id="posterior">
        <prior id="prior">
            <logNormalPrior mean="-15.08" stdev="0.625" offset="0.0">
                <parameter idref="clock.rate_Medi"/>
            </logNormalPrior>
            <uniformPrior lower="0.0" upper="1.5E6" offset="0.0">
                <parameter idref="treeModel.rootHeight_Pama"/>
            </uniformPrior>
            <gammaPrior shape="1.0" scale="1.0" offset="0.0">
                <parameter idref="siteModel.alpha_Medi"/>
            </gammaPrior>
            <coalescentLikelihood idref="coalescent_Medi"/>
            <coalescentLikelihood idref="coalescent_Pama"/>
            <coalescentLikelihood idref="coalescent_Eugr"/>
        </prior>
```

```
        ...
```

Here is a simple example of the entire MCMC element:

```
<mcmc id="mcmc" chainLength="1000000" autoOptimize="true">
    <posterior id="posterior">
        <prior id="prior">
            <distributionLikelihood idref="distributionLikelihood"/>
            <speciationLikelihood idref="speciation"/>
        </prior>
        <likelihood id="likelihood">
            <treeLikelihood idref="treeLikelihood"/>
        </likelihood>
    </posterior>

    <operators idref="operators"/>

    <log id="screenLog" logEvery="500">
        <column label="Likelihood" dp="4" width="12">
            <posterior idref="posterior"/>
        </column>
        <column label="Root Height" sf="4" width="12">
            <parameter idref="treeModel.rootHeight"/>
        </column>
    </log>

    <log id="fileLog" logEvery="100" fileName="example.log">
        <posterior idref="posterior"/>
        <parameter idref="siteModel.alpha"/>
        <parameter idref="siteModel.pInv"/>
        <parameter idref="gtr1.ac"/>
        <parameter idref="gtr1.ag"/>
        <parameter idref="gtr1.at"/>
        <parameter idref="gtr1.cg"/>
        <parameter idref="gtr1.gt"/>
        <parameter idref="yule.birthRate"/>
        <rateStatistic idref="meanRate"/>
        <rateStatistic idref="rateVariance"/>
        <rateStatistic idref="rateCoeff"/>
        <rateCovarianceStatistic idref="covariance"/>
        <parameter idref="treeModel.rootHeight"/>
        <tmrcaStatistic idref="hominins-mrca"/>
        <tmrcaStatistic idref="primates-mrca"/>
        <tmrcaStatistic idref="rodents-mrca"/>
    </log>

    <logTree id="treeFileLog" logEvery="1000" nexusFormat="true" fileName="example.trees">
        <treeModel idref="treeModel"/>
    </logTree>
</mcmc>
```

4.2.19 Other elements

For help on other aspects of the BEAST XML including:

- Setting up two epoch models
- Running BEAST without data in order sample from the joint prior distribution
- Using nucleotide substitution models other than HKY or GTR and

- Information on the general data type

see the BEAST homepage or `http://beast.bio.ed.ac.uk/BEAST_XML_Reference` for more details.

5 TRACER

Tracer is a simple piece of software that can be used for visualization and diagnostic analysis of the MCMC output of BEAST. It reads BEAST (and MrBayes) log files. As with BEAUti and BEAST the exact instructions for running TRACER differ depending on the type of computer Tracer is being used on. Once it is running, however, TRACER will look similar irrespective of which operating system it is running on.

5.1 Importing log files into Tracer

Once Tracer has been opened you will see in the top left hand corner the "Trace Files" panel. Below that will be a "+" and "-" button. To load your log output files press on the + button. A file dialog will appear allowing to choose a log file to load. Multiple log files can be loaded into Tracer this way and will be displayed in the "Trace Files" panel. By selecting multiple log files, you can click the combine button to combine these log files for further analysis. Combining log files is only appropriate if they represent independent replicates of the same BEAST analysis. The "Trace Files" panel will also display the number of generations that the MCMC algorithm ran for and the burn-in period (set by default to 10% of the MCMC chain length when a log file is loaded into Tracer). You can double click on the burn-in value to change them if visual inspection of the trace suggests that 10% is not appropriate.

5.2 Analysis using Tracer

On the left hand side of the Tracer window is the name of the log file loaded and the parameters/statistics that it contains. There will usually be a trace of the posterior (this is the sum of the log likelihood of the tree, the log prior probability of the tree and the log prior probability density of any other priors). There will also be traces of the continuous parameters such as `hky.kappa` and `treeModel.rootHeight`. Selecting a trace on the left brings up a statistical summary for the trace on the right hand side depending on the tab selected (see below). For each trace the mean value and the Effective Sample Size (ESS) will also be displayed. If the ESS is red (it is flagged red if less than 100) then the MCMC chain has not been run long enough to get a valid estimate of the parameter. The ESS is an estimate of how many effectively independent draws from the marginal posterior distribution the MCMC is equivalent to. If this number is small then the log file may not accurately represent the posterior distribution and more (or longer) MCMC runs need to be run (see below for more details).

The basic statistics available for each trace are:

Mean — The mean value of the sampled trace across the chain (excluding values in the burn-in).

Stdev	The standard error of the estimated mean, taking into account the ESS, so a small ESS will give a large standard error (stdev of the mean).
Median	The median value of the sampled trace across the chain (excluding the burn-in).
95% HPD Lower	The lower bound of the 95% highest posterior density (HPD) interval. The 95% HPD is shortest interval that contains 95% of the sampled values.
95% HPD Upper	The upper bound of the highest posterior density (HPD) interval. The 95% HPD is shortest interval that contains 95% of the sampled values.
Auto-correlation time	The number of states in the MCMC chain that two samples have to be separated by on average to be regarded as independent samples from the posterior distribution. The smaller the ACT the better the MCMC chain is mixing. The ACT is estimated from the samples in the trace (excluding the burn-in).
Effective Sample Size	The ESS is the number of independent samples from the marginal posterior distribution that the trace is equivalent to. It is calculated by dividing the chain length (excluding the burn-in) by the ACT.

You can select a trace and look at the raw trace in the "Trace" panel. This is the most important step in Tracer analysis. If the Trace for each parameter has not converged on a stationary distribution (i.e., it looks like a straight hairy caterpillar with no obvious upward or downward trends or sudden jumps) then the MCMC run needs to be run for longer. Once the MCMC chain has been run for long enough the frequency histogram will generally be a smooth unimodal distribution (although this doesnt have to be the case). Selecting a trace and displaying it in a density plot will show the posterior probability density of a parameter. Running the MCMC chain longer can reduce the stochastic noise in all the plots. You can also select multiple parameters (especially if you have partitioned data and look at them on the same trace).

5.3 Increasing Effective Sample Size

For publication purposes we recommend that all ESS values be greater than 200. If the ESS is small then the estimate of the posterior distribution of that parameter will be poor and the standard error of the mean of parameter will be large. Low ESS values are indicative of poor mixing and should cast doubt on the validity of all parameter estimates in the log file. Trying for ESSs values greater than 1000 is probably a waste of computational resources. Parameters that have high ESSs can have their operators down-weighted; whereas parameters with low ESSs may need their operators weights to be increased.

There are a number of ways to increase the ESSs of your parameters:

- Increase the chain length. This is the most straightforward way of increasing the ESS.

- Combine results from multiple independent runs. We recommend that you do multiple runs of your analysis and compare results to check that the chains are converging and mixing adequately, thus they should be sampling the same distribution and results could be combined (once a suitable burn-in is removed from each run). The continuous parameters can be analysed and combined using Tracer. Tree files have to be combined manually using a text editor.

- Optimize the performance of the operators by using the auto-optimize option in the Operator panel in BEAUti.

- Increase the sample frequency (if you have less than 1000 samples in total). This may help because the ESS is measuring the correlation between sampled states in the chain. If the sample frequency is very low each sample will be independent and ESS will be approximately equal to the number of states in the log file. Therefore increasing the sample frequency will increase the ESS. Sampling too frequently will not increase the ESS but will increase the size of the log file and the time it takes to analyse it. A balance of these two considerations suggests that sampling so that the total number of samples is between 1,000 and 10,000 is recommended.

6 LogCombiner

LogCombiner allows you to combine log and tree files from multiple independent runs of BEAST. When this program is opened the LogCombiner user interface and a JAVA LogCombiner window will appear.

6.1 File Type

This combo menu allows you to select either the log or tree file type that you will be importing into LogCombiner.

6.2 Resample states at lower frequency

This option allows you to resample your posterior distribution at a lower frequency than in previous BEAST runs.

6.3 Select input files

Here you can select using the "+" button the input files that you wish to combine. These will appear in the sub-window with the file name and the BurnIn peroid (by default 10%).

6.4 Output file

This option allows you to select a log file or create a new log file that the combined log data will be saved to.

When you click "Run", the log files you have selected will be combined in the JAVA LogCombiner window. The files you have selected must be from independent runs of BEAST from the same XML file, otherwise an error will

occur stating that the number of columns in the first file does not match that of the second file. Once LogCombiner has finished you can analyse the combined log file in Tracer.

Important: It does not make sense to combine log files from MCMC analyses of different models or different data sets.

7 TreeAnnotator

This program assists in summarizing the information from a sample of trees produced by BEAST onto a single "target" tree. The summary information includes the posterior probabilities of the nodes in the target tree, the posterior estimates and HPD limits of the node heights and (in the case of a relaxed molecular clock model) the rates.

7.1 BurnIn

This option allows you to select the amount of burn-in, i.e., the number of samples that will be discarded at the start of the run, so that you are only analysing the part of the trace that is in equilibrium.

7.2 Posterior probability limit

This is the same as specifying a a limit for bootstrapping in PAUP*. Posterior summaries will only be calculated for the nodes in the target tree that have a posterior probability greater than the specified limit.

7.3 Target tree type

If you select the "Maximum clade credibility" option then the node height and rate statistics will be summarized on the tree in the posterior sample that has the maximum **sum** of posterior probabilities on its $n - 2$ internal nodes. This tree is not necessarily the majority-rule consensus tree.

If you select the "User target tree" then the tree statistics will be summarized on a user-specified tree. This could, for example, be a majority-rule consensus tree constructed from the posterior tree sample using PAUP*.

7.4 Node heights

This option allows you select how the node heights are summarised on the target tree. You can choose to keep the heights that the target tree has, or rescale it to reflect the posterior mean/median node heights for the clades contained in the target tree.

7.5 Target tree file

This option allows you to select and input the target tree. This option will only be available if you have selected "User target tree" from the "Target tree type" combo menu.

7.6 Input tree file

This option allows you to select the input tree file (the tree file produced by a BEAST analysis).

7.7 Output file

This option allows you to select or create a file that the summarized tree data will be saved to.

Once you click run, TreeAnnotator will start to summarize the tree data produced by BEAST. In the JAVA window TreeAnnotator will state the number of trees that have been read, will find the best fit tree specified under the "Target tree type". The progress of this will be monitored by the * symbols moving across the screen. TreeAnnotator will also give you a clade support statistic before writing the annotated tree to file. This file can then be analysed in FigTree.

Authors Contributions

AJD and AR designed and implemented all versions of BEAST up to the current (version 1.4.2), which was developed between June 2002 and April 2007. Portions of the BEAST source code are based on an original Markov chain Monte Carlo program developed by AJD (called MEPI) during his PhD at Auckland University between the years 2000 and 2002. Portions of the BEAST source code are based on previous C++ software developed by AR. BEAST is now an open source project and many software developers have made invaluable contributions (see Acknowledgements for some of them). SYWH produced the initial version of the BEAST XML guide as an online resource. AJD twisted the arm of NR to bring together the disparate sources of information on BEAST into a first draft of this manual. All authors contributed to the writing of the text.

Acknowledgements

We would like to thank (in alphabetical order) Roald Forsberg, Joseph Heled, Philippe Lemey, Gerton Lunter, Sidney Markowitz, Oliver G. Pybus, Tulio de Oliveira, Beth Shapiro, Korbinian Strimmer and Mark A. Suchard for invaluable contributions. AR was supported by the Royal Society.

References

[1] Huelsenbeck JP, Ronquist F: MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* 2001, **17**:754-755.

[2] Drummond AJ, Nicholls GK, Rodrigo AG, Solomon W: Estimating mutation parameters, population history and genealogy simultaneously from temporally spaced sequence data. *Genetics* 2002, **161**(3):1307-1320.

[3] Wilson IJ, Weale ME, Balding DJ: Inferences from DNA data: population histories, evolutionary processes and forensic match probabilities. *J Royal Stat Soc A-Statistics in Society* 2003, **166**:155-188.

[4] Beaumont MA: Detecting population expansion and decline using microsatellites. *Genetics* 1999, **153**(4):2013-2029.

[5] Rannala B, Yang ZH: Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics* 2003, **164**(4):1645-1656.

[6] Pybus OG, Drummond AJ, Nakano T, Robertson BH, Rambaut A: The epidemiology and iatrogenic transmission of hepatitis C virus in Egypt: a Bayesian coalescent approach. *Mol Biol Evol* 2003, **20**(3):381-387.

[7] Kuhner, MK: LAMARC 2.0: Maximum likelihood and Bayesian estimation of population parameters. *Bioinformatics* 2006 **22**(6):768-770.

[8] Redelings BD, Suchard MA: Joint Bayesian Estimation of Alignment and Phylogeny. *Syst Biol* 2005, **54**(3):401-418.

[9] Lunter G, Miklos I, Drummond A, Jensen JL, Hein J: Bayesian coestimation of phylogeny and sequence alignment. *BMC Bioinformatics* 2005, **6**(1):83.

[10] Hastings WK: Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 1970, **57**:97-109.

[11] Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E: Equations of state calculations by fast computing machines. *J Chem Phys* 1953, **21**:1087-1091.

[12] Zuckerkandl E, Pauling L: Evolutionary divergence and convergence in proteins. In: *Evolving genes and proteins*. Edited by Bryson V, Vogel HJ. Academic Press: New York; 1965: 97-166.

[13] Aris-Brosou S, Yang Z: Bayesian models of episodic evolution support a late Precambrian explosive diversification of the Metazoa. *Mol Biol Evol* 2003, **20**(12):1947-1954.

[14] Kishino H, Thorne JL, Bruno WJ: Performance of a divergence time estimation method under a probabilistic model of rate evolution. Molecular Biology & Evolution 2001, 18:352-361.

[15] Sanderson MJ: Estimating absolute rates of molecular evolution and divergence times: A penalized likelihood approach. *Mol Biol Evol* 2002, **19**:101-109.

[16] Thorne JL, Kishino H: Divergence time and evolutionary rate estimation with multilocus data. *Syst Biol* 2002, **51**(5):689-702.

[17] Thorne JL, Kishino H, Painter IS: Estimating the rate of evolution of the rate of molecular evolution. *Mol Biol Evol* 1998, **15**:1647-1657.

[18] Yoder AD, Yang ZH: Estimation of primate speciation dates using local molecular clocks. *Mol Biol Evol* 2000, **17**:1081-1090.

[19] Suchard MA, Redelings BD: BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. *Bioinformatics* 2006 **22**(16):2047-2048.

[20] Rambaut A, Drummond AJ: Tracer [computer program] Available from http://evolve.zoo.ox.ac.uk/software/ 2003

[21] Shapiro B, Drummond AJ, Rambaut A, Wilson MC, Matheus PE, Sher AV, Pybus OG, Gilbert MT, Barnes I, Binladen J et al: Rise and fall of the Beringian steppe bison. *Science* 2004, **306**(5701):1561-1565.

[22] Rodriguez F, Oliver JL, Marin A, Medina JR: The general stochastic model of nucleotide substitution. *J Theor Biol* 1990, **142**(4):485-501.

[23] Hasegawa M, Kishino H, Yano T: Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol* 1985, **22**(2):160-174.

[24] Goldman N, Yang Z: A codon-based model of nucleotide substitution for protein-coding DNA sequences. *Mol Biol Evol* 1994, **11**(5):725-736.

[25] Yang Z: Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. *J Mol Evol* 1994, **39**(3):306-314.

[26] Drummond AJ, Pybus OG, Rambaut A, Forsberg R, Rodrigo AG: Measurably evolving populations. *Trends Ecol Evol* 2003, **18**(9):481-488.

[27] Griffiths RC, Tavare S: Sampling theory for neutral alleles in a varying environment. *Philos Trans R Soc Lond B Biol Sci* 1994, **344**(1310):403-410.

[28] Kingman JFC: The coalescent. *Stochastic Processes and Their Applications* 1982, **13**:235-248.

[29] Drummond AJ, Rambaut A, Shapiro B, Pybus OG: Bayesian coalescent inference of past population dynamics from molecular sequences. *Mol Biol Evol* 2005, **22**(5):1185-1192.

[30] Aldous DJ: Stochastic models and descriptive statistics for phylogenetic trees, from Yule to today. *Statistical Science* 2001, **16**(1):23-34.

[31] Drummond AJ, Ho SYW, Phillips MJ, Rambaut A: Relaxed phylogenetics and dating with confidence. *PLoS Biology* 2006, **4**(5)

[32] Thorner JL, Kishino H, Felsenstein J: An evolutionary model for maximum likelihood alignment of DNA sequences. *J Mol Evol* 1991, **33**(2): 114-124.

[33] Lemey P, Pybus OG, Rambaut A, Drummond AJ, Robertson DL, Roques P, Worobey M, Vandamme AM: The molecular population genetics of HIV-1 group O. *Genetics* 2004, **167**(3):1059-1068.