Model adequacy using BEAST2

Assessing clock and substitution models

David A. Duchêne

1 Background

This tutorial will guide you through methods to assess model adequacy in BEAST v2.4.X. In common practice, evolutionary models are selected based on their statistical fit *relative to each other*. This might be sufficient in some cases, but there is a risk that all of the candidate models lead to inferences that poorly represent the true evolutionary process. Indeed, even the most complex or best fitting model from a set of candidates can produce highly erroneous estimates of parameters of interest. In this tutorial we will explore methods to investigate the absolute merits of the model. This kind of assessment of the absolute performance of models is also known as model checking or assessment of model adequacy or plausibility.

Before starting the tutorial, it is important that you understand the methods used in Bayesian inference for assessing model adequacy. A typical assessment of model adequacy is done by comparing a test statistic, calculated from the empirical data set, with the values of the statistic calculated from a large number of data sets simulated under the model. The simulated data sets from the model are often referred to as posterior predictive simulations (PPS), and they represent future or alternative data sets under the candidate model. The test statistic should be informative about the assumptions of the model in question.

A large number of test statistics have been proposed to assess the components of phylogenetic analyses. In this tutorial we will investigate two test statistics, one for assessing the substitution model, and one for assessing the priors on rates and times used for molecular dating (Figure 1).

- The multinomial likelihood, which is the likelihood of the data under a model with only a single general assumption: that substitution events are independent and identically distributed across sites. This statistic can be used to assess overall substitution model fit. Note that sites with missing data or indels should not be included when estimating the multinomial likelihood.
- The A index assesses the power of the molecular-clock model to estimate the number of substitutions across branches, assuming a tree topology and an adequate substitution model. We will go through this statistic in more detail during this exercise.

2 Programs used in this Exercise

- BEAST v2.4.X.
- R programming environment.
- R packages *ape* and *phangorn*. Before starting the exercise, install these packages by typing the following in your R console:

```
install.packages("phangorn")
```

3 Run the empirical data

In the `data` folder you will find a simulated sequence alignment with 2000 nucleotides in nexus format (`cp3.nex`) and a Yule-process chronogram in newick format (`corvid.tre`) with 50 taxa. The alignment was simulated along the chronogram under a Jukes-Cantor substitution model and an **uncorrelated log-normal clock model**. You will also find an XML file (`cp3.xml`) in the `xml` folder to run BEAST 2 for this data set.

The evolutionary history of birds has been widely studied using mitochondrial DNA. Mitochondrial genomes have been sequenced for a diversity of birds, including passerines (perching birds). However, can show substantial differences in nucleotide composition between species. This potentially violates an assumption that is commonly made in phylogenetic analyses: that of compositional homogeneity across lineages.

When we analyse nucleotide sequences, we often compare a number of substitution models from the GTR family to identify the one that provides the best fit to our data. However, we typically only consider a small subset of the models in this family, such as the JC, HKY, and GTR models. But it is possible that all of the models in this subset actually provide a poor absolute fit to our data.

Here we investigate whether the widely used GTR+G model of nucleotide substitution provides an adequate description of the evolutionary process in mitochondrial DNA from 13 corvid birds (plus an outgroup taxon). This group of birds includes crows, ravens, and the Eurasian magpie. The data set comprises 1000 sites from the third codon positions of mitochondrial protein-coding genes.

The settings in this XML file include:

- A GTR+G substitution model.
- A **strict clock** with a uniform prior from 0 to 1 (substitution per site per million years).
- A yule tree prior.
- A normally distributed root calibration with standard deviation of 2 and mean of 40.
- Run length of 10 million steps and the sampling frequency as 1K.

Importantly, this XML file has been modified directly such that the starting tree has been provided and it remains fixed throughout the run. The steps to modify the XML to fix the tree topology are explained in a box at the end of this tutorial.

Run the XML file using BEAST 2. The output of this run can also be found in the folder `precooked_runs`.

4 One step model assessment

In this section we will run substitution and clock model assessment. Although we will be using the output from the BEAST 2 analysis that we ran in the previous section, **you should use this section and that for model interpretation if you wish to perform model assessment using your own data.**

The results for the example data set in this section can also be found in the folder `precooked_runs`. In that folder you will also find the results for the BEAST 2 run of `cp3.xml`.

Begin by opening R. By typing the following, you will set the working directory to the scripts folder, and then load all the necessary scripts inside it.

```
setwd("INSERT THE PATH TO SCRIPTS FOLDER")
```

```
for(i in dir()) source(i)
```

In the next box will set the directory to `precooked_runs`, and run the function `adeq()`. The information in quotes in `adeq()` are:

- The path to the posterior of trees from BEAST 2
- The path to the log file from BEAST 2
- The path to the empirical data alignment in nexus format

In addition, we indicate `Nsim` = 100, which indicates the number of posterior predictive simulations to be performed.

```
setwd("../precooked_runs")
clock_adequacy_example <- adeq(trees.file = "cp3.trees", log.file ="cp3.log", empdat.file = "↩
    ../data/cp3.nex", Nsim = 100)
names(clock_adequacy_example)
```

The contents of `clock_adequacy_example` should appear after the latest line of code. The contents include each of the components used to assess clock and substitution model adequacy. Elements that have the word `empirical` are test statistics calculated for empirical data. Elements that say `pps` are data for each of the simulated data sets.

The following are some detailed elements in `clock_adequacy_example`:

- The seventh element `empirical_multlik` is the multinomial likelihood calculated for the empirical data set.
- The eighth element `pps_multliks` are the values of the multinomial likelihood calculated for each of the simulated data sets.
- The ninth element `mlik_pvalue` is the *P*-value for the multinomial likelihood, a proxy for the distance between the simulations and the empirical data.

The `clock_adequacy_example` object should have the same contents as object "assessment_provided" in the file `results.Rdata`:

```
load("results.Rdata")
names(assessment_provided)
```

The following section of this tutorial describes the steps for assessing model adequacy in more detail.

5 Steps for assessing model adequacy

5.1 Reading the runs and simulating data

We will study the code in R and Figure 1 in detail. This section is mainly for reading and discussion, and is aimed for you to cement the steps required for assessing model adequacy.

Open the `adeq.R` file in a text editor of your preference and you should see the following:

```
adeq <- function(trees.file, log.file, empdat.file, Nsim = 100){
    empdat <- as.phyDat(as.DNAbin(read.nexus.data(empdat.file)))
    seqlen <- ncol(as.matrix(as.DNAbin(empdat)))
    tree.topo <- read.nexus(trees.file)[[1]]
    sims <- make.pps.als(trees.file, log.file, Nsim, seqlen)
    sims <- make.pps.tr(sims, empdat, tree.topo)
    bls <- compile.results(sims)
    return(bls)
}
```

The analysis of empirical data appears in blue in Figure 2. We then need to read the posterior trees and parameter estimates of the BEAST 2 analysis. The R package *phangorn* allows us to take these data and run the posterior predictive simulations, which is shown in green in Figure 1. You will find the code to read data from the posterior and for simulating genetic alignments in `make.pps.als`. This script identifies the model being assessed and simulates data accordingly. The input required in this step includes:

- The paths of the posterior files for your analysis.
- The number of simulations you want to perform.
- The sequence length (number of sites in your alignment).

If you are interested in how we read and simulate data in R, you can investigate the `make.pps.als` code. The following is the line in `adeq.R` in question:

```
    sims <- make.pps.als(trees.file, log.file, Nsim, seqlen)
```

If the input file has a greater number of samples than the number of simulations requested, this will randomly select samples from the posterior. This is a good moment to discuss or read about how an alignment can be simulated using data from the posterior.

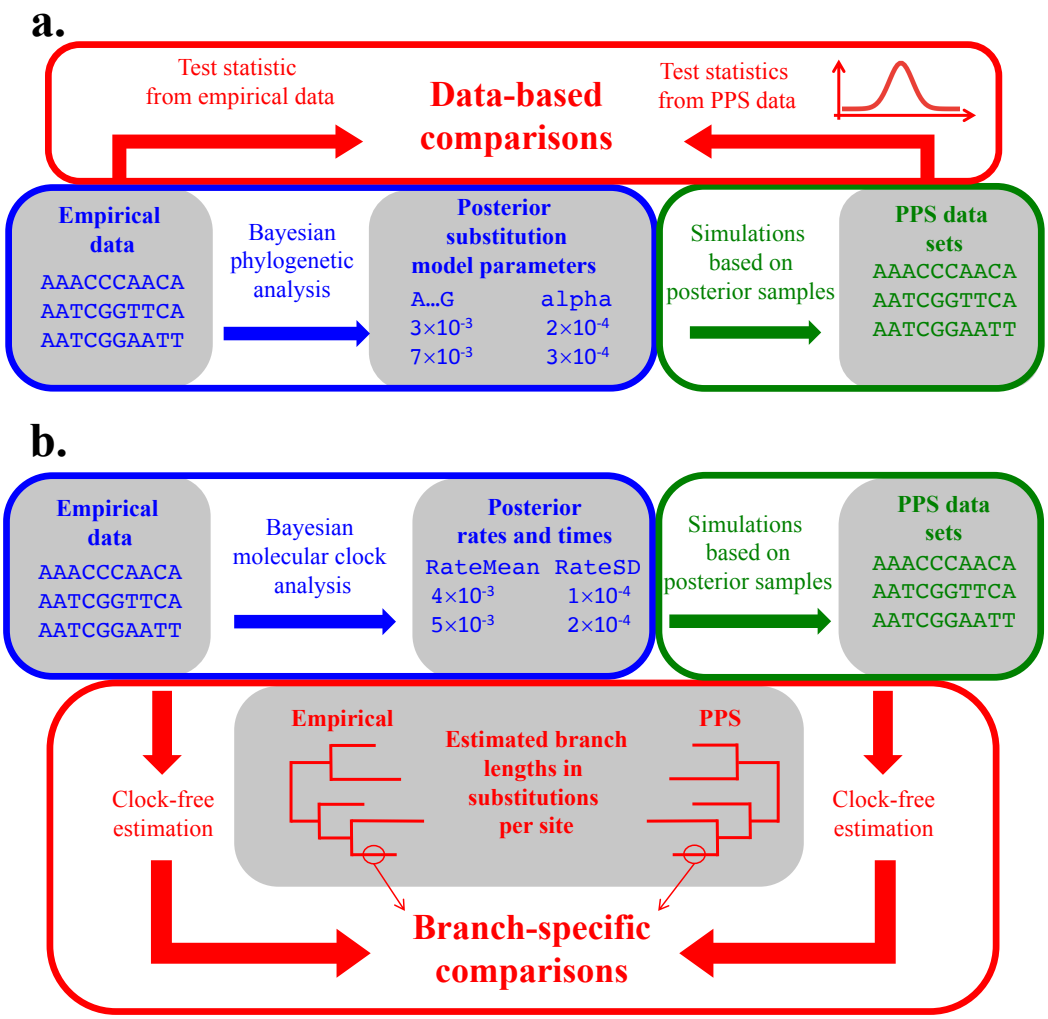

Figure 1: Two of the existing approaches to using posterior predictive simulations to assess model adequacy in Bayesian phylogenetics. (a) One group of methods use characteristics of the data for model assessment, like the multinomial likelihood or the GC content. (b) Another method can assess clock models using estimates from clock-free methods. Under this approach, the number of substitutions per site expected along each branch under the clock hierarchical model are compared with those inferred in a clock-free analysis of the empirical data.

5.1.1 Calculate test statistics

Once we have simulated data sets, we can calculate the test statistics. The function `make.pps.trs` takes the assumed tree, substitution model, and the empirical and simulated data sets. The function estimates phylogenetic branch lengths for the empirical data set and each of the simulated data sets. In this step we also estimate the multinomial likelihood test statistic for the empirical data and each of the simulated data sets.

```
sims <- make.pps.tr(sims, empdat, tree.topo)
```

The output of this function is what we need for model assessment: the test statistics for the empirical data, and the distribution of test statistics for the simulated data sets.

5.1.2 Calculate *P*-values

We can now compare the test statistic for the empirical data and each of the simulated data sets, which is the step shown in red in Figure 1. The most common way to do this is to calculate the tail area probability, which is the number of simulations with a test statistic greater than the value for the empirical data.

```
bls <- compile.results(sims)
```

This function will provide the test statistics for simulations, as well as *P*-values for each of the test statistics. Following practice from frequentist statistics, we can consider the model to be inadequate if the *P*-value for a given test statistic is below 0.05. Importantly, the assessment of the clock model allows us to identify the branches for which the molecular clock model can estimate the number of substitutions. We will explore the interpretation of these data in the following section.

6 Interpreting the results of model assessment

6.1 Results of substitution model assessment

It is strongly recommended to use qualitative checks of models using graphical analyses. This section uses the results in `precooked_runs/results.Rdata` and **not** those from the previous steps. If you want to use your results you need to replace `assessment_provided` with the name of your results (`clock_adequacy_example`). In this section we will graph different components for assessing clock model adequacy using posterior predictive simulations.

We will first visualise the results for assessing substitution model adequacy. The following code makes a histogram of the distribution of the multinomial likelihood for the PPS data, and will show the position of the value for empirical data on this distribution.

```
hist(assessment_provided[[8]],
     xlim = c(min(assessment_provided[[8]])-sd(assessment_provided[[8]]),
              max(assessment_provided[[8]])+sd(assessment_provided[[8]])),
     main = "", xlab = "Multinomial likelihood")

abline(v = assessment_provided[[7]], col = 2, lwd = 3)
```

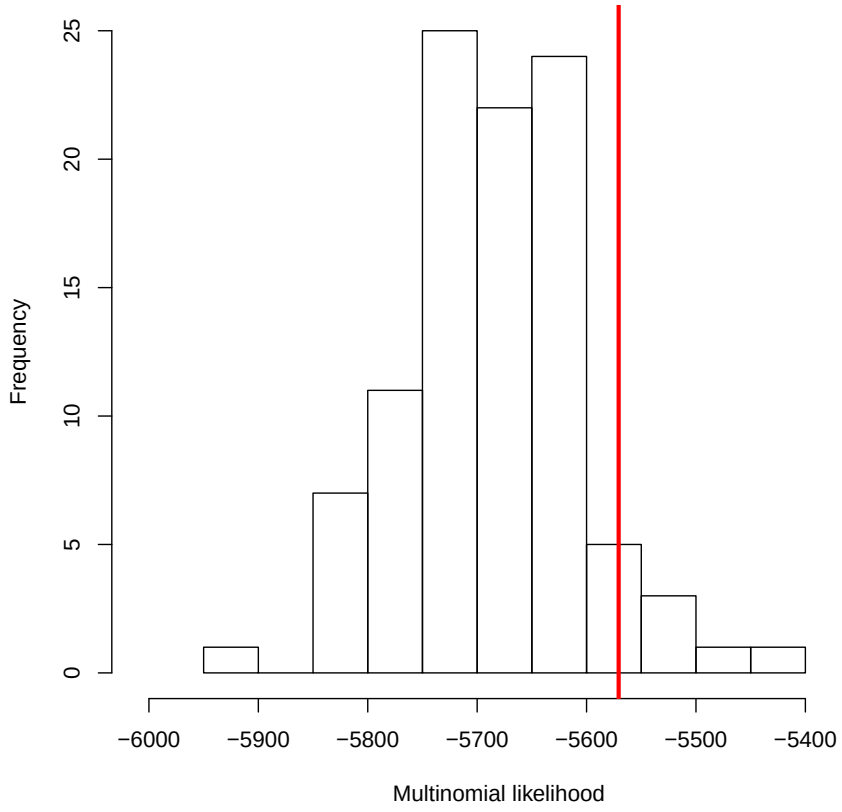

Figure 2: Distribution of PPS multinomial likelihood values with the value of the test statistic for the empirical data shown as a vertical line in red.

Your plot will be identical or very similar to Figure 2. When assessing model adequacy, we consider the model to be an adequate representation of the evolutionary process if the test statistic for the empirical data is a typical value arising from the model. The multinomial likelihood for the empirical data falls inside the distribution of values for simulated data (Figure 2), so our substitution model is possibly a reasonable description of the process that generated the data.

This result can also be observed in the P-value for the multinomial likelihood in R:

```
assessment_provided[9]
```

6.1.1 Results of clock model assessment

The following script shows a simple example to explore the branch-wise posterior predictive P-values. We will first load the tree. In this example we will use the original tree provided, but usually the tree with the median posterior branching times would be appropriate. We will colour the branches with the best accuracy in blue, and the branches that have the lowest accuracy in green (Figure 3).

```
tr <- read.tree("../data/corvid.tre")
plot(tr, edge.col = rainbow(length(assessment_provided$branch_wise_pppvalues),
     start = 2/6, end = 4/6)[rank(assessment_provided$branch_wise_pppvalues)],
     edge.width = 6, cex = 1.5)
edgelabels(assessment_provided$branch_wise_pppvalues, bg = "white", cex = 1.5, frame = "none")
```

The values along each branch indicate the proportion of simulations in which the branch-length was greater than the length estimated using the empirical data. The expected value under the model is 0.5. If this value is 0 branch-lengths are being underestimated with respect to the model. Similarly, if the value is 1 branch-lengths are being overestimated with respect to the model.

You can also investigate the A index:

```
assessment_provided[4]
```

This index is the proportion of branches in the tree for which the branch-wise posterior predictive P-values are inside the central 95 percent of the distribution. The rates and times models can be considered adequate when the A index is high.

You might find it surprising that the A index in these data are closer to 0 than 1. The reason for this is possibly that we used an overly simple model of rate variation across lineages. As described in section 3, the analyses in BEAST 2 were done using a strict clock. Furthermore, these data have very high overall rates, such that they have a very large amount of variation, and possibly contain substantial substitutional saturation.

The following script shows a simple example to explore the branch-wise length deviation. This metric is a proxy for the difference between the branch-length estimate using empirical data and the mean branch-length estimate using posterior predictive simulations. To make the values comparable across branches, the difference between the empirical branch length and mean PPS branch-length has been divided by the empirical branch length. If this value is close to zero, the priors for times and molecular evolutionary rates can be considered adequate. We apply the same colouring system as the plot above. Note that in the case of branch length deviation, larger numbers indicate greater deviation from the empirical branch length, and therefore lower accuracy (Figure 4).

Have a look at the alignment file to observe the large amount of variation in the data, and why some of the values in Figure 4 seem extremely large. Our conclusion is that the strict clock provides a very poor fit to these data, and a more flexible clock model might be better suited.

```
plot(tr, edge.col = rev(rainbow(length(assessment_provided$branch_length_deviation),
     start = 2/6, end = 4/6)[rank(assessment_provided$branch_length_deviation)]),
     edge.width = 6, cex = 1.5)
edgelabels(round(assessment_provided$branch_length_deviation, 2), bg = "white", cex = 1.5, ↩
    frame = "none")
```

Note that in this simple method to graph the results, the branches in the two plots above have been coloured by their rank, rather than their magnitude.

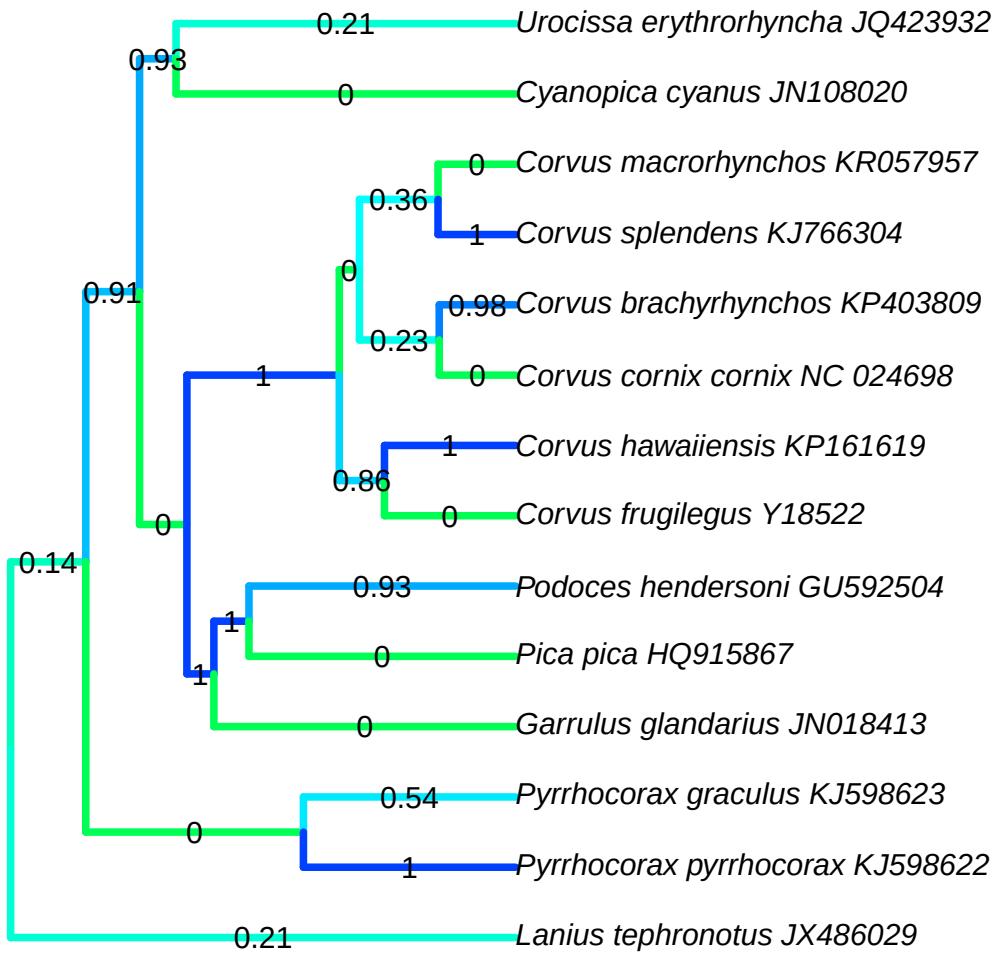

Figure 3: Estimated chronogram with branches coloured by their clock adequacy P-value.

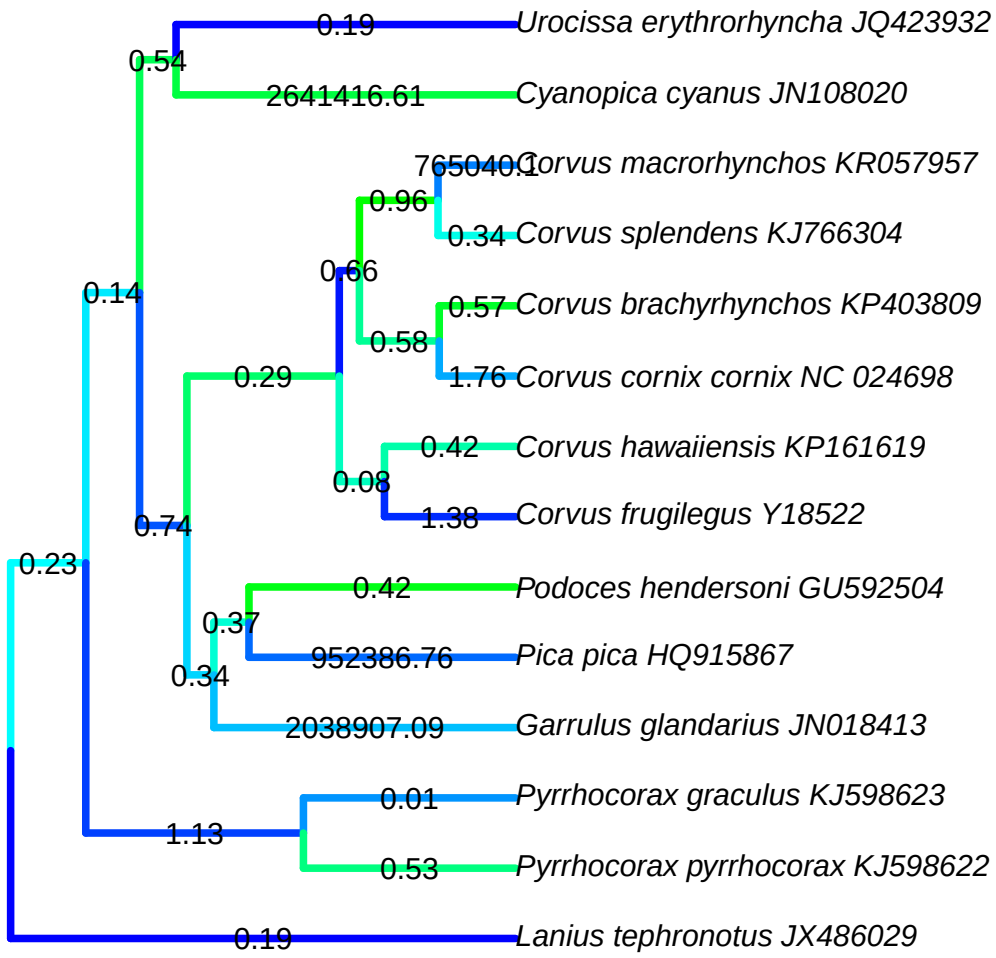

Figure 4: Estimated chronogram with branches coloured by their deviation between the empirical and simulated lengths.

7 Concluding remarks

- Assessing model adequacy provides insight into the *absolute* merits of a candidate model. This means that it can identify cases when even the best-fitting candidate model is a poor representation of the evolutionary process, such that inferences are unreliable. Assessment of model adequacy is an alternative to traditional model selection approaches that assess the *relative* support among candidate models.
- The choice of test statistic is critical to the assessment of model adequacy. The test statistic can define what aspect of a model is being assessed. Most test statistics are not complete descriptions of the model, so it is convenient to explore a model using several test statistics where possible.
- There are cases in which the inferences of some parameters are reliable, while others are not (e.g. some branch rates are accurate while others are not). Closer examination of test statistics can sometimes bring additional insight into when the model is adequate for the data at hand.

Optional: Fixing the tree topology

Load the data `cp3.nex` to BEAUti, setting the model as described in the first section, and saving the XML file. Then open the XML file in your preferred text editor, and add a `tree` statement as follows above line with the `<run ...` statement.

```
<tree id="Tree.t:treename" spec='beast.util.TreeParser' newick="((A, B), (C, D), E);" ↩
    taxa="@alignmentname"/>
```

Newick refers to a format to express the tree structure. Replace the dummy newick tree with the correct tree, which you will find in `corvid.tre` in the `data` folder. Also modify the names of the tree and the taxa: the name of your tree appears after `"Tree.t:` elsewhere in the XML, and the name of the taxa is an `@` followed by your data `id` in the line immediately above the alignment.

Now remove the `tree` statement, which is inside the `state` and looks like the following.

```
<tree id="Tree.t:treename" name="stateNode">
        <taxonset id="TaxonSet.alignmentname" spec="TaxonSet">
            <alignment idref="alignmentname"/>
        </taxonset>
 </tree>
```

In the place of that `tree` statement, type an `input` statement, which will lead to a new `statenode` as follows.

```
<input name='stateNode' idref='Tree.t:treename'/>
```

Once again, make sure that the tree name is correct.

Now remove the whole random tree initialiser, which begins and ends with the following lines.

```
<init id="RandomTree.t:treename" ...
...
</init>
```

Lastly, remove the lines (or set the weights to 0) of the operators on the tree topology. These include the lines that have any of `SubtreeSlide`, `Narrow`, `Wide`, and `WilsonBalding`. You might find these words together with the tree model selected (in this case `BirthDeath`).

Version dated: February 9, 2017