Babette practical

Thijs Janzen

2024-09-14

Inferring phylogenies

There are many softwares to infer phylogenies from a sequence. Here, in this practical, we will make use of the R package 'babette', which provides an interface to make use of the software BEAST2 (Bayesian Evolutionary Analysis Sampling Trees). Although BEAST2 by default provides an interface to setup a phylogenetic analysis, this interface can be tricky to work with, and it can be hard to replicate analyses made using this interface. Instead, Richel Bilderbeek has developed the R package 'babette', which allows us to write an R script to perform a BEAST2 analysis.

Installing required packages

To be able to perform our analyses, there are a number of packages that need to be installed, including of course BEAST2. Let's get started!

```r
install.packages("ape")
install.packages("beastier")
install.packages("tracerer")
install.packages("mauricer")
install.packages("beautier")

# this installs beast2:
remotes::install_github("richelbilderbeek/beastierinstall")
beastierinstall::install_beast2()

install.packages("babette")

# and for plotting:
install.packages("tidyverse")
```

Now, we have all packages installed. Unfortunately, BEAST2 needs Java to run. This can turn out to be quite finnicky to get to work for some laptops. Please first try:

```r
install.packages("rJava")
```

If that does not work to install rJava, please also see these notes: https://github.com/ropensci/babette/blob/master/doc/rjava.md

Please note that getting rJava and Java to run can, in some cases (e.g. with older laptops), be a complicated process. If you don't manage to get Java to run, please don't spend hours trying to get this to work, but instead see if you can join forces with a neighbor or fellow-student that did manage to get everything correctly installed.

All packages can now be loaded:

```r
library(tidyverse)
library(babette)
library(ape)
library(beautier)
library(mauricer)
library(tracerer)
```

Loading data

We can load data into R using the following functions, where we load an example file from the BEAST2 examples (alternatively, download the file from Brightspace). We then use the package ape to read the example file.

```r
nexus_filename <- beastier::get_beast2_example_filename("dna.nex")

nexus_data <- ape::read.nexus.data(nexus_filename)
output_data <- ape::as.DNAbin(nexus_data)

ape::write.FASTA(output_data, file = "dna.fas")

str(nexus_data)
```

```
## List of 10
##  $ Cow    : chr [1:705] "a" "t" "g" "g" ...
##  $ Carp   : chr [1:705] "a" "t" "g" "g" ...
##  $ Chicken: chr [1:705] "a" "t" "g" "g" ...
##  $ Human  : chr [1:705] "a" "t" "g" "g" ...
##  $ Loach  : chr [1:705] "a" "t" "g" "g" ...
##  $ Mouse  : chr [1:705] "a" "t" "g" "g" ...
##  $ Rat    : chr [1:705] "a" "t" "g" "g" ...
##  $ Seal   : chr [1:705] "a" "t" "g" "g" ...
##  $ Whale  : chr [1:705] "a" "t" "g" "g" ...
##  $ Frog   : chr [1:705] "a" "t" "g" "g" ...
```

The last command shows us that the dataset we have obtained containes sequences of 10 quite different species. Our goal in this tutorial is to infer their relationships and find when they diverged from each other.

Setting up babette analysis

To set up a babette analysis, we have to specify a number of prerequisites #### Tree prior To set up the tree prior, e.g. we predefine a model of how we expect species to branch over time, we can make use of the predefined tree priors in beautier, such as: Yule, Birth-Death and different coalescent priors (see ?beautier::create_tree_priors for an overview of all available priors).
Here, we will make use of a birth-death tree prior.

```r
bd_prior <- beautier::create_tree_prior_bd()
```

Clock model The clock model translates genetic differences into time units. In other words, it determines how 'fast' the clock ticks. However, typically we find that the clock does not tick equally fast along all branches, and this needs to be corrected for. Instead, typically it is assumed that the clock rates of each branch are slightly different, but all belong to the same distribution. Babette currently supports two clock models: the strict clock model, that assumes clock rates along all branches are identical, and the relaxed log normal clock model, that assumes clock rates along all branches are drawn from a log normal distribution. We will use the latter.

```
rln_clock <- beautier::create_clock_model_rln()
```

Substitution model The substition model chooses how bases mutate into each other. In babette, there are a number of substitution models available, such as the JC69, HKY, TN93 and GTR model. The simplest model is the JC69 model; the JC69 model assumes that all bases have equal probability of mutating into each other, e.g. a mutation from A to T has the same probability as a mutation from A to G. On the other end of the extreme, The Generalised time reversible (GTR) model fits all 6 different substitution rates for each possible substitution (please verify that you can capture all possible substitutions between 4 bases, with 6 rates). The other available models pose some restrictions on the potential rates, and fall in between the JC69 and GTR model in terms of complexity. In this tutorial we will use the JC69 model for simplicity.

```
sub_model <- beautier::create_site_model_jc69()
```

Define MCMC BEAST2 makes use of Markov Chain Monte Carlo (MCMC) to find the best fitting parameters. MCMC is a numerical method aimed at stochastically (the Monte Carlo part) exploring parameter space, weighed by the fit. In other words: we aim to explore most the area with the best fit, and not spend time in parameter space with poor fit. Ideally, we would map the entirety of parameter space to find the perfect fit, however given the number of parameters this is impossible, and instead we will make use of MCMC. The core idea is that we pick a parameter combination and calculate the likelihood, then, we slightly perturb the parameter combination and calculate the likelihood again. Depending on the improvement of the likelihood, we move to the new parameter combination or not. Iteratively doing so creates a parameter 'chain', where each entry only depends on the entry before (e.g. a Markov Chain). Thus moving through parameter space is an efficient way of exploring only those areas with high likelihood. However, values are often auto-correlated, and to avoid this, a common practice is to only store every Nth parameter combination. Here, we will use an interval of 1000, and generate a chain of 1e6 combinations:

```
mcmc_settings <- beautier::create_mcmc(chain_length = 1e6, store_every = 1000)
```

Running Babette

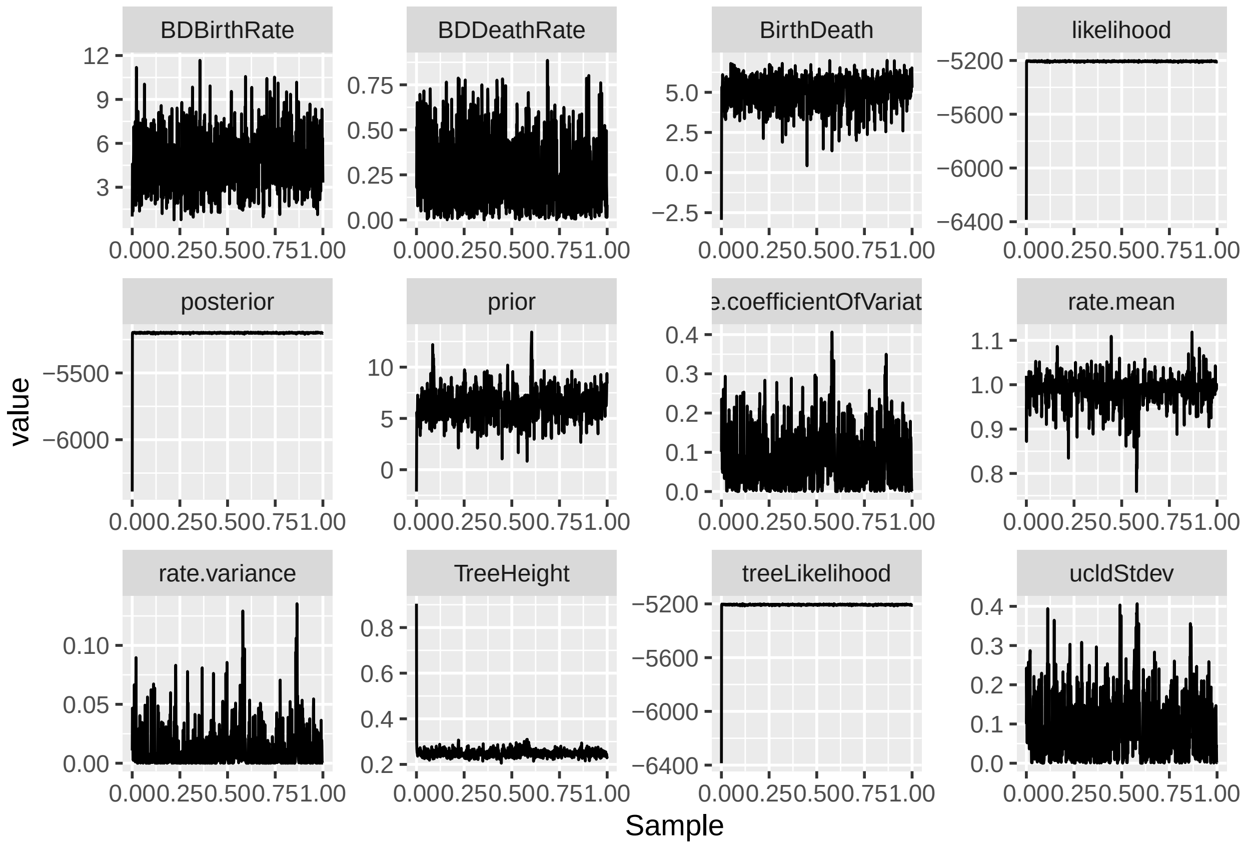
Now we can combine our priors and mcmc settings into an inference model, and pass this inference model to babette to start the inference

```
inf_model <- beautier::create_inference_model(tree_prior = bd_prior,
                                              site_model = sub_model,
                                              clock_model = rln_clock,
                                              mcmc = mcmc_settings)

beast_result <- babette::bbt_run_from_model(fasta_filename = "dna.fas",
                                            inference_model = inf_model,
                                            beast2_options = beastier::create_beast2_options(rng_seed =
```

Analysing results

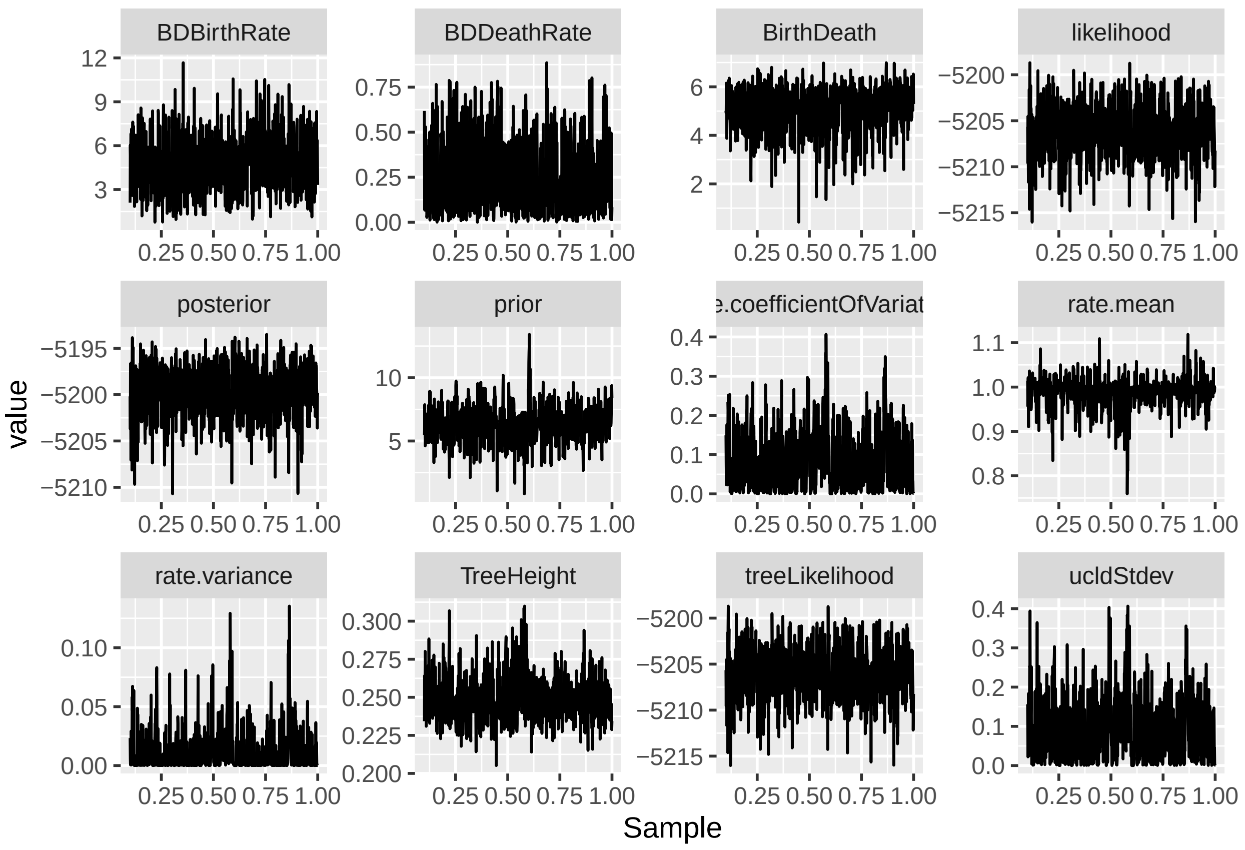
Now that we have our results, we can plot them and see how we did:

```r
beast_result$estimates %>%
  mutate("Sample" = Sample / 1000000) %>%
  gather(key = "statistic", val = "value", -c(Sample)) %>%
  ggplot(aes(x = Sample, y = value)) +
  geom_line() +
  facet_wrap(~statistic, scales = "free")
```



In these plots, we see on the x-axis our sample (these are one thousand points, right?), and on the y-axis the associated value. Shown are results for 12 different optimized parameters. If our Markov Chain converged well, we should see 'Hairy Caterpillars' for all parameters, indicating that the chain was circling around the optimum and sampling well. Looking at our results, we immediately see that a number of plots show very different results in the beginning, these are the result of initialization of the chain. This is a common occurance, and is typically removed from the dataset as 'burn in'. Here, we will remove the first 10% of the samples to get rid of these poor datapoints. Typically, removal of 10-20% of the data is considered acceptable.

```r
beast_result$estimates <- tracerer::remove_burn_ins(beast_result$estimates, 0.1)
beast_result$estimates %>%
  mutate("Sample" = Sample / 1000000) %>%
  gather(key = "statistic", val = "value", -c(Sample)) %>%
  ggplot(aes(x = Sample, y = value)) +
  geom_line() +
  facet_wrap(~statistic, scales = "free")
```



Upon re-plotting the data, we see that now, all variables have turned into hairy caterpillars. A further check of whether our inference was long enough is by calculating the Effective Sample Size (ESS), this should be at least 200 for each variable:

```
tracerer::calc_esses(beast_result$estimates, sample_interval = 1000)
```

```
##     posterior likelihood prior treeLikelihood TreeHeight ucldStdev rate.mean
## 101       657        648   265            648        197       206       210
##     rate.variance rate.coefficientOfVariation BirthDeath BDBirthRate
## 101           190                         215        555         701
##     BDDeathRate
## 101         549
```

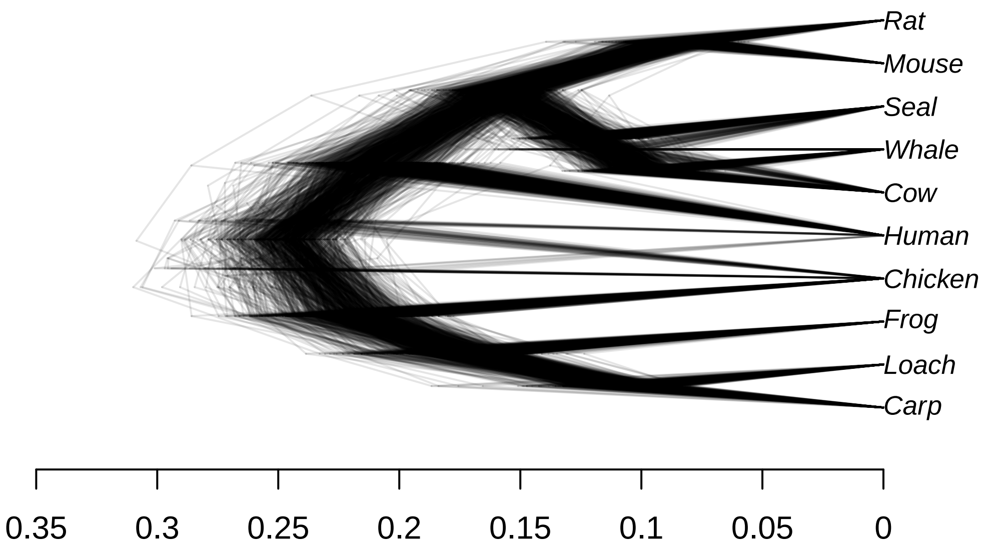
For some variables (TreeHeight and rate.variance) values are not 200 yet. The best way forward would be to either repeat the analysis with a longer chain, or continue the existing chain. For the purpose of this tutorial, we will continue with the existing results, also considering values are already quite close to 200.

The MCMC results have also generated 1000 trees. Of these, again we have to remove the first 10%, and then we can plot these trees to see whether they have converged unto a unifying tree.

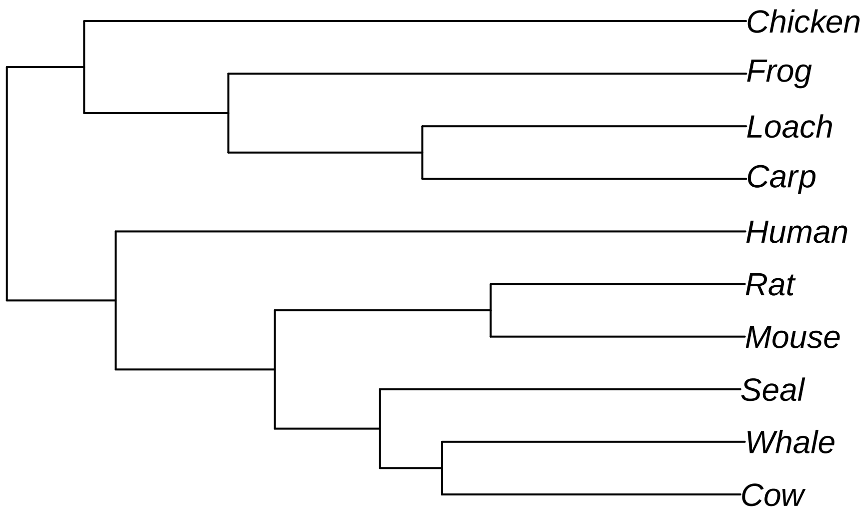
```
all_trees <- beast_result$dna_trees
start <- floor(0.1 * length(all_trees))
end <- length(all_trees)
all_trees <- all_trees[start:end]
```

Having removed the burnin, we can now plot our results in two ways: 1) using a densi-tree, which superimposes all trees on top of each other and 2) using a consensus tree, which calculates the 'average' resulting tree from our distribution of 900 trees.

```r
babette::plot_densitree(all_trees, alpha = 0.1)
```



```r
cons_tree <- phytools::consensus.edges(trees = all_trees)
plot(cons_tree)
```



In the consensus tree, we see a topology that seems to make sense - species that are close related appear close to each other. However, the consensus tree does not tell us anything about uncertainty in the tree estimation process. The densitree does. And looking at the densitree, we see some interesting patters, mainly regarding the root, where some trees in the posterior have swapped the position of humans and chicken.

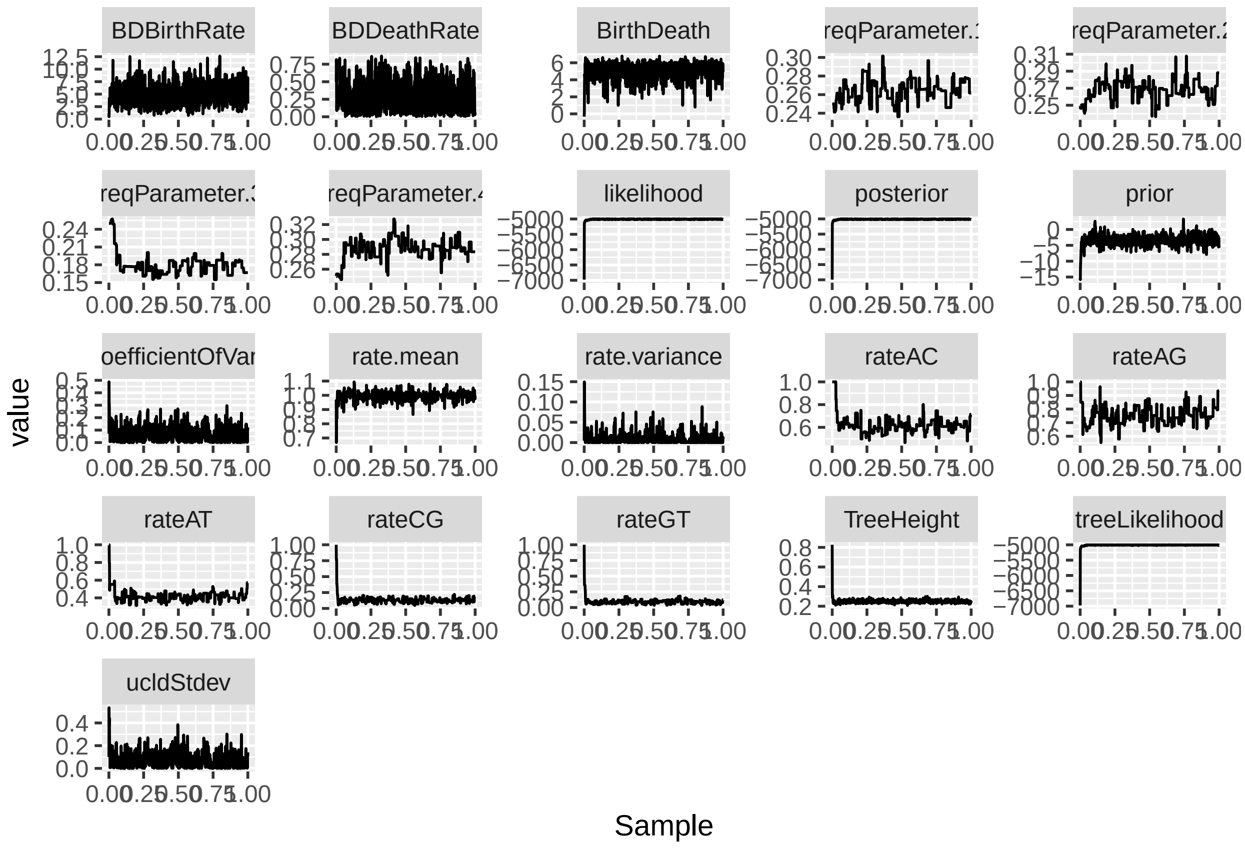
Substitution model

How does the substitution model affect our results? We can easily repeat the analysis, but now change the substitution model. Let's do so!

```r
sub_model <- beautier::create_site_model_gtr()

inf_model <- beautier::create_inference_model(tree_prior = bd_prior,
                                              site_model = sub_model,
                                              clock_model = rln_clock,
                                              mcmc = mcmc_settings)

beast_result_gtr <- babette::bbt_run_from_model(fasta_filename = "dna.fas",
                                            inference_model = inf_model,
                                            beast2_options = beastier::create_beast2_options(rng_seed =
```

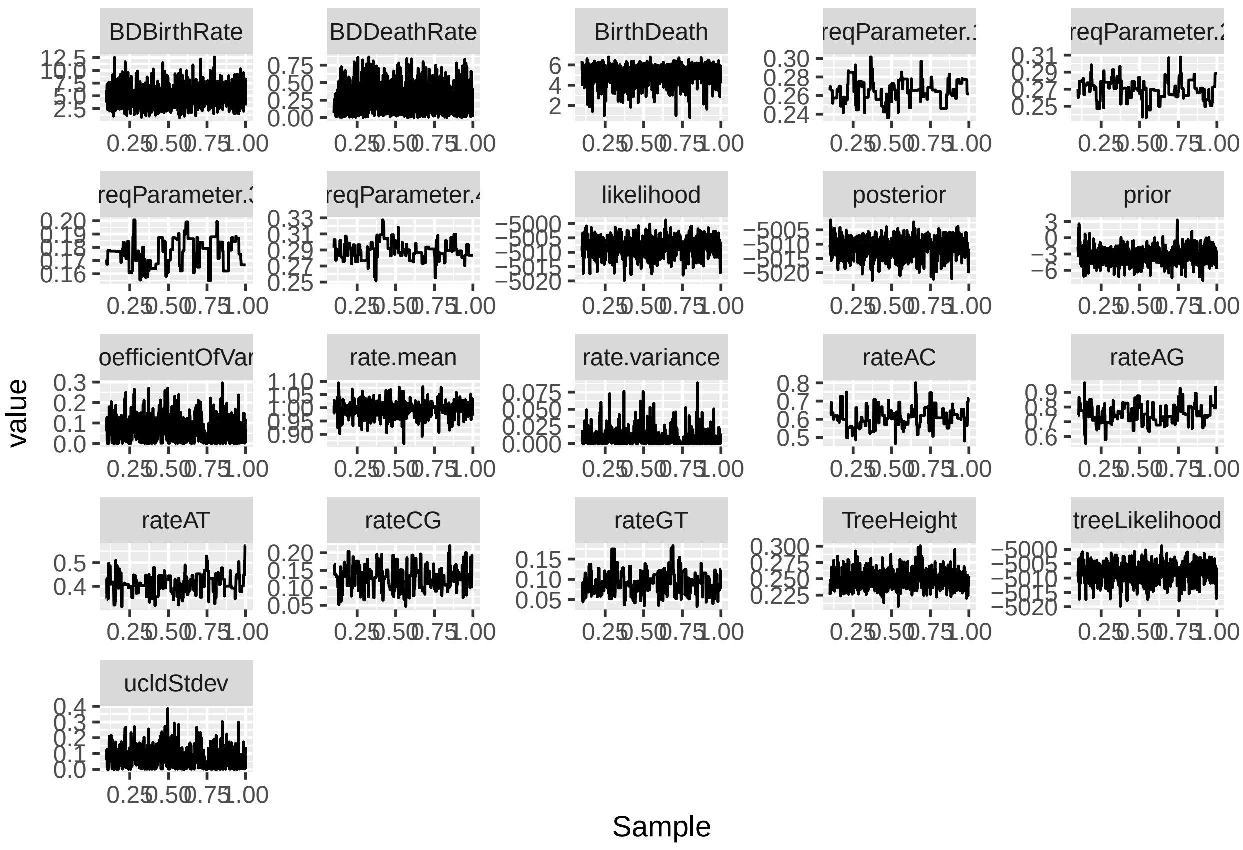
If we plot our results, we will see that additional variables have been added to reflect estimates for the different substitution rates:

```
beast_result_gtr$estimates %>%
  mutate("Sample" = Sample / 1000000) %>%
  gather(key = "statistic", val = "value", -c(Sample)) %>%
  ggplot(aes(x = Sample, y = value)) +
  geom_line() +
  facet_wrap(~statistic, scales = "free")
```



This is not yet very informative due to the burnin effects. Also, perhaps it is interesting to look at the median estimates for the different rates:

```
gtr_estimates <- tracerer::remove_burn_ins(beast_result_gtr$estimates, 0.1)
gtr_estimates %>%
  mutate("Sample" = Sample / 1000000) %>%
  gather(key = "statistic", val = "value", -c(Sample)) %>%
  ggplot(aes(x = Sample, y = value)) +
  geom_line() +
  facet_wrap(~statistic, scales = "free")
```



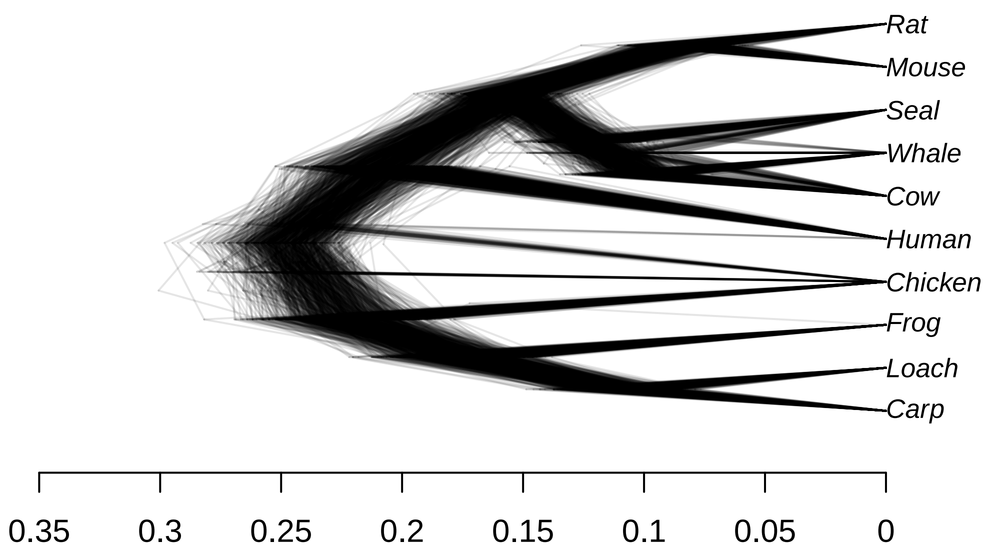
```
gtr_estimates %>%
  gather(key = "statistic", val = "value", c(rateAC, rateAG, rateAT, rateCG, rateGT)) %>%
  group_by(statistic) %>%
  summarise("median" = median(value))
```

```
## # A tibble: 5 x 2
##   statistic median
##   <chr>      <dbl>
## 1 rateAC    0.607
## 2 rateAG    0.748
## 3 rateAT    0.408
## 4 rateCG    0.132
## 5 rateGT    0.0923
```

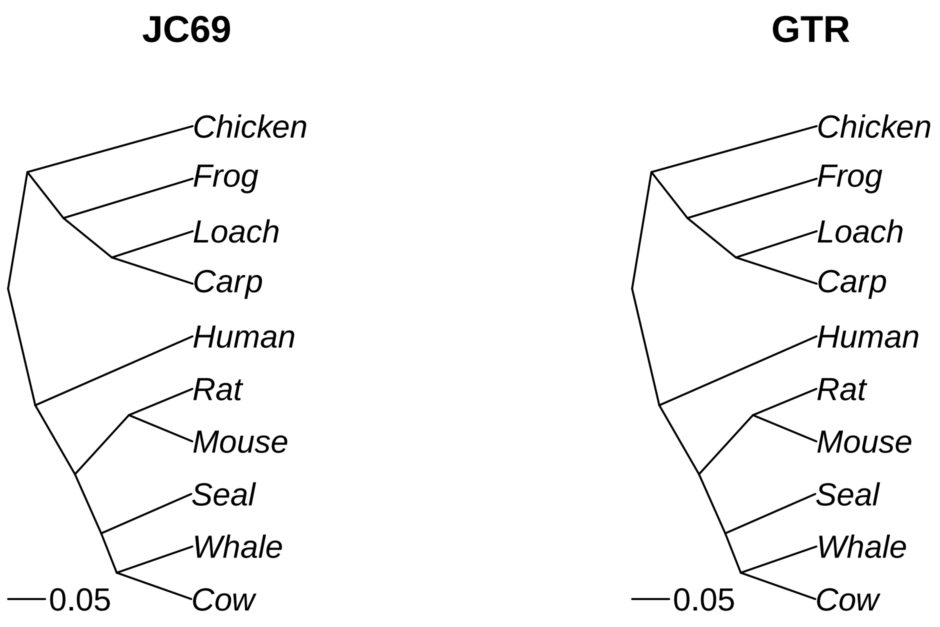
From these values, it is clear that substitutions from A to C (0.6) and A to G (0.75) where estimated to be most frequent, with substitutions from A to T (0.41), C to G (0.13) and G to T (0.09) much less frequent. For the JC69 model we had no rate estimates, as there all substitutions are equally likely (so the rate is '1').

Does a different substitution model affect our findings? We collect again the trees after removal of burnin, and then plot the tree side by side with the JC69 findings. Also, we add a plot where we plot the lineages over time to see if our substitution model has changed anything.

```
gtr_trees <- beast_result_gtr$dna_trees
start <- floor(0.1 * length(gtr_trees))
end <- length(gtr_trees)
gtr_trees <- gtr_trees[start:end]
babette::plot_densitree(gtr_trees, alpha = 0.1)
```



```
cons_tree_gtr <- phytools::consensus.edges(trees = all_trees)
par(mfrow = c(1, 2))
plot(cons_tree, main = "JC69", type = "cladogram")
add.scale.bar()
plot(cons_tree_gtr, main = "GTR", type = "cladogram")
add.scale.bar()
```



It seems the information in the data is very clear, and that choices about the substitution model do not impact our findings.

Root age

It is clear from the densitree, that the root age has been inferred incorrectly: the root is now placed at 0.25 MY, which seems very, very young. Indeed, birds and mammals are thought to have diverged from each other around 319 MYA (sd = 1.64) (timetree.org). We can add this information into our analysis by putting a prior on the crown age.

```
mrca_prior <- beautier::create_mrca_prior(
     taxa_names = beautier::get_taxa_names("dna.fas"),
     is_monophyletic = TRUE,
     mrca_distr = beautier::create_normal_distr(mean = 319, sigma = 1.64))
```

Now we have to re-specify our inference model to include the mrca prior, and because we are using an mrca prior, we also have to force usage of beauti 2.6. Lastly, because we are adding complexity to the inference, we expect the chain to take longer to converge. To anticipate this, we have chosen a longer chain length.

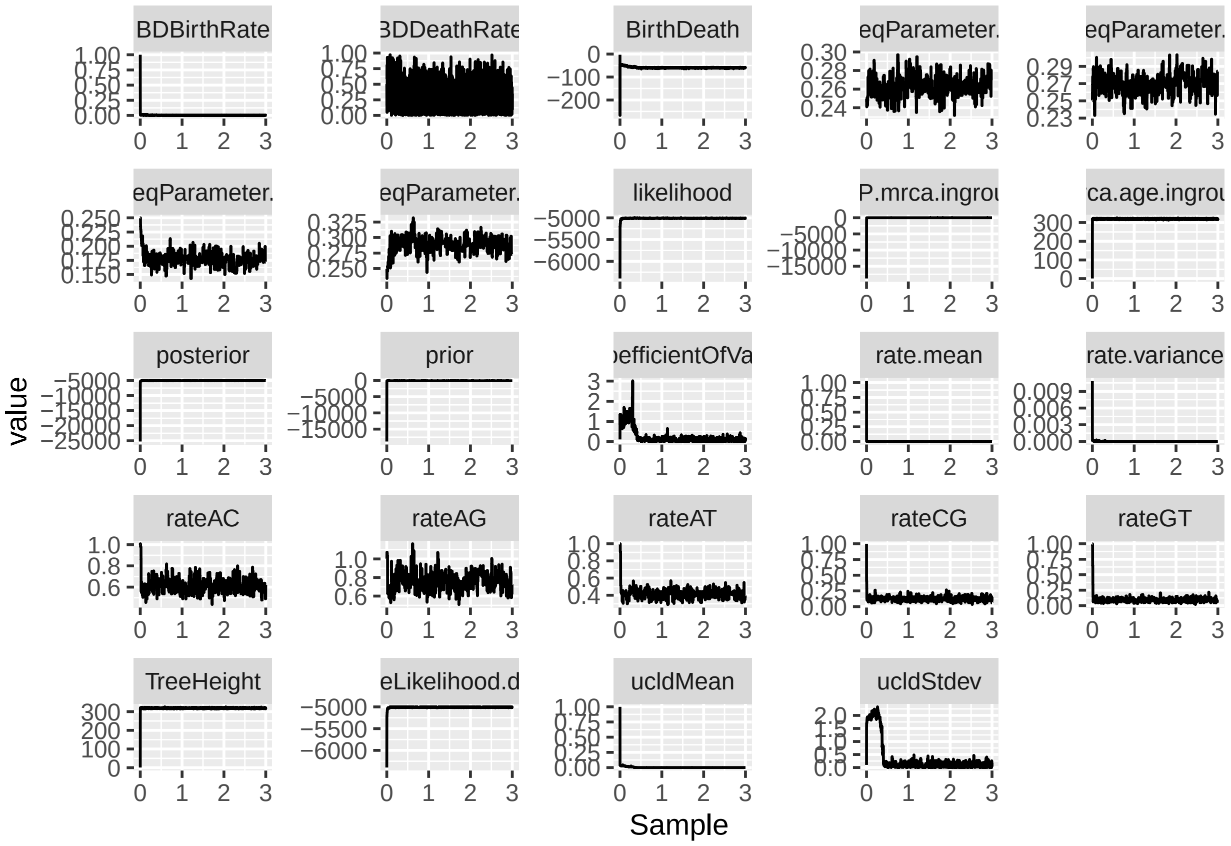
```
mcmc_settings <- beautier::create_mcmc(chain_length = 3e6, store_every = 1000)


beauti_options <- beautier::create_beauti_options_v2_6()

inf_model <- beautier::create_inference_model(tree_prior = bd_prior,
                                              clock_model = rln_clock,
                                              site_model = sub_model,
                                              mcmc = mcmc_settings,
                                              mrca_prior = mrca_prior,
                                              beauti_options = beauti_options)
```

And now we can repeat the analysis:

```
beast_result <- babette::bbt_run_from_model(fasta_filename = "dna.fas",
                                            inference_model = inf_model,
                                            beast2_options = beastier::create_beast2_options(rng_seed =
```

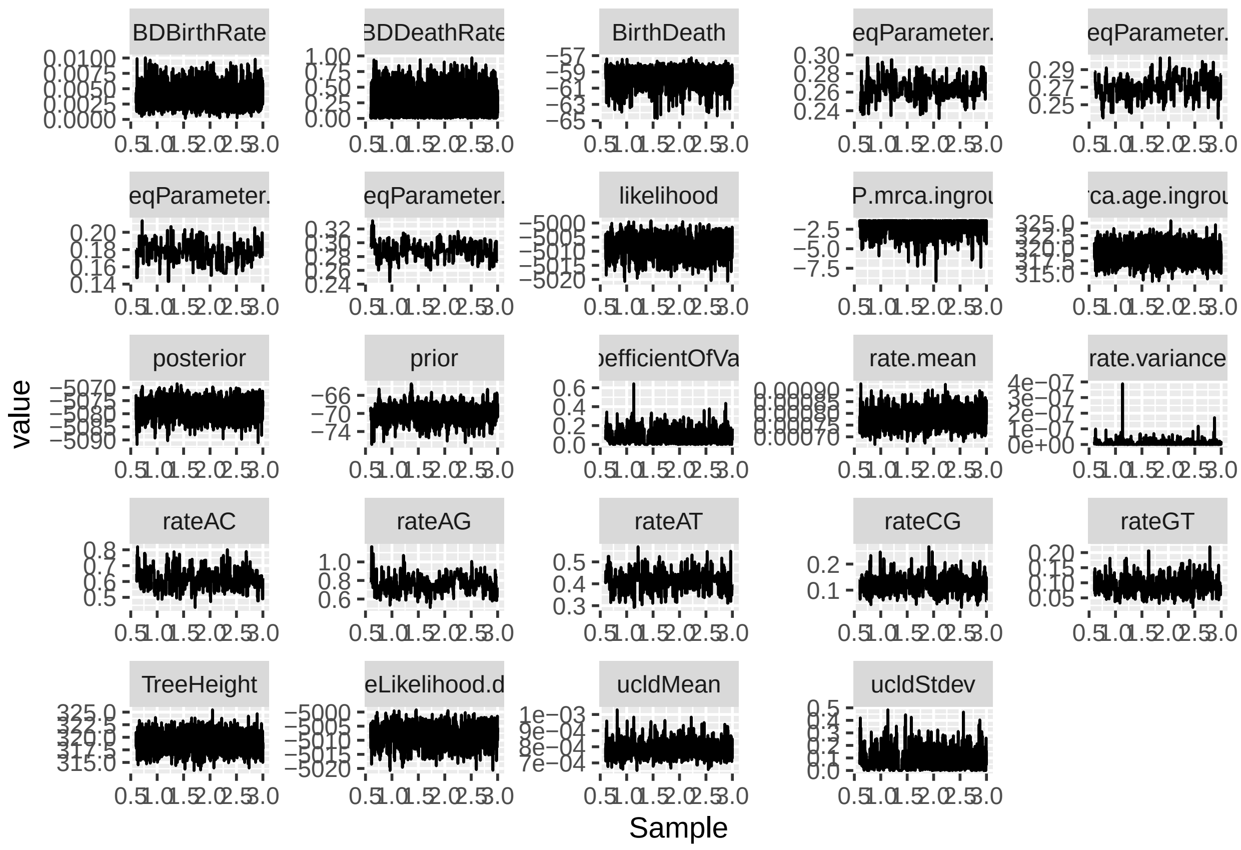
Again, we first plot our MCMC chains:

```
beast_result$estimates %>%
  mutate("Sample" = Sample / 1000000) %>%
  gather(key = "statistic", val = "value", -c(Sample)) %>%
  ggplot(aes(x = Sample, y = value)) +
  geom_line() +
  facet_wrap(~statistic, scales = "free")
```



Clearly, we have strong initialization effects. Let's remove 20% this time:

```
estimates <- tracerer::remove_burn_ins(beast_result$estimates, 0.2)
estimates %>%
  mutate("Sample" = Sample / 1000000) %>%
  gather(key = "statistic", val = "value", -c(Sample)) %>%
  ggplot(aes(x = Sample, y = value)) +
  geom_line() +
  facet_wrap(~statistic, scales = "free")
```



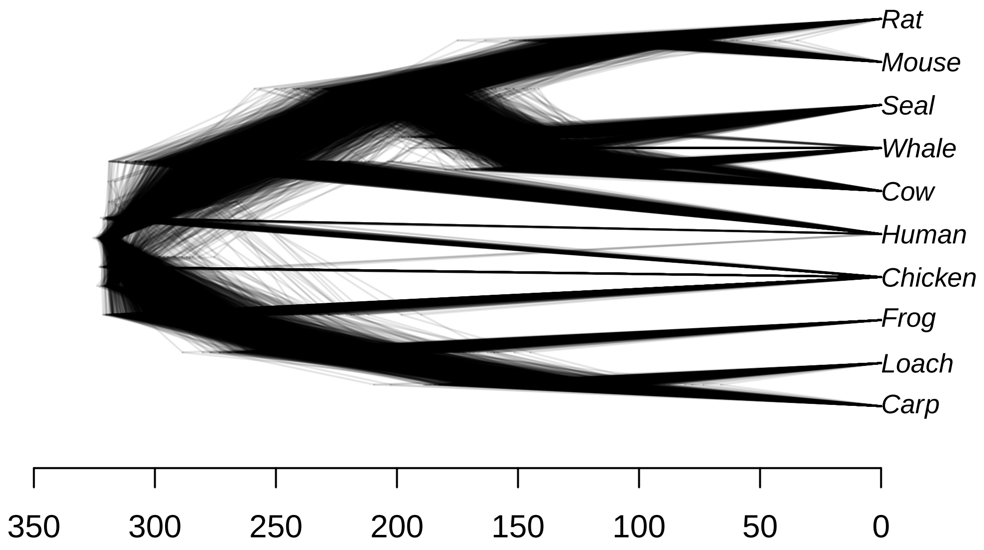
```
tracerer::calc_esses(estimates, sample_interval = 1000)
```

```
##     posterior likelihood prior treeLikelihood.dna TreeHeight rateAC rateAG
## 601       343        387   202                387       2143    120     45
##     rateAT rateCG rateGT freqParameter.1 freqParameter.2 freqParameter.3
## 601    201    243    155             148             104              40
##     freqParameter.4 ucldMean ucldStdev rate.mean rate.variance
## 601             105      419       197       640           326
##     rate.coefficientOfVariation BirthDeath BDBirthRate BDDeathRate
## 601                         195       1986        1655        1604
##     logP.mrca.ingroup.. mrca.age.ingroup.
## 601                2401              2143
```

Having lengthened the chain now pays off, as almost all ESS values are > 200.

We again remove the burn-in trees as well, and plot the distribution of trees:

```
all_trees <- beast_result$dna_trees
start <- floor(0.2 * length(all_trees))
end <- length(all_trees)
all_trees <- all_trees[start:end]

babette::plot_densitree(all_trees, alpha = 0.1)
```



Now, we see that the crown of the tree has moved towards 320MY, and that the other branching points have moved accordingly. Our tree looks much better now!

Outlook

We have made a number of simplifying assumptions, and we have also left some settings at their default. For instance, we have, in this tutorial, not looked into setting a prior on the substitution model. To have a look at other potential priors you can set and options you can explore, please have a look at the help page for the function 'beautier::create_inference_model' for further information.

Exercise

Now that you've got a feel for the analysis pipeline, it is your turn! Please load DNA sequences for 12 primate species, and reconstruct their phylogeny. Can you also infer the age of the most recent common ancestor of Humans and Chimpansee? If you have time: how does the substition model influence your findings? Why / Why not?

You can load the DNA sequences from the beast2 examples as follows:

```r
nexus_filename <- beastier::get_beast2_example_filename("Primates.nex")
fasta_filename <- "primates.fas"

nexus_data <- ape::read.nexus.data(nexus_filename)
output_data <- ape::as.DNAbin(nexus_data)

ape::write.FASTA(output_data, file = fasta_filename)
str(nexus_data)
```

```
## List of 12
##  $ Tarsius_syrichta: chr [1:898] "a" "a" "g" "t" ...
##  $ Lemur_catta     : chr [1:898] "a" "a" "g" "c" ...
##  $ Homo_sapiens    : chr [1:898] "a" "a" "g" "c" ...
##  $ Pan             : chr [1:898] "a" "a" "g" "c" ...
##  $ Gorilla         : chr [1:898] "a" "a" "g" "c" ...
##  $ Pongo           : chr [1:898] "a" "a" "g" "c" ...
##  $ Hylobates       : chr [1:898] "a" "a" "g" "c" ...
##  $ Macaca_fuscata  : chr [1:898] "a" "a" "g" "c" ...
##  $ M_mulatta       : chr [1:898] "a" "a" "g" "c" ...
##  $ M_fascicularis  : chr [1:898] "a" "a" "g" "c" ...
##  $ M_sylvanus      : chr [1:898] "a" "a" "g" "c" ...
##  $ Saimiri_sciureus: chr [1:898] "a" "a" "g" "c" ...
```

Once again: if you can not get rJava to work, please do not spend hours to fix this, but rather see if your neighbor has been able to install rJava. Then, work together to finish this exercise.

Good luck!