babette: BEAUti 2, BEAST2 and Tracer for R

Richèl J.C. Bilderbeek

Rampal S. Etienne

**1.** In the field of phylogenetics, BEAST2 is one of the most widely used software tools. It comes with the graphical user interfaces BEAUti 2, DensiTree and Tracer, to create BEAST2 configuration files and to interpret BEAST2’s output files. However, when many different alignments or model setups are required, a workflow of graphical user interfaces is cumbersome.  
**2.** Here, we present a free, libre and open-source package, babette: ’BEAUti 2, BEAST2 and Tracer for R’, for the R programming language. babette creates BEAST2 input files, runs BEAST2 and parses its results, all from an R function call.  
**3.** We describe babette’s usage and the novel functionality it provides compared to the original tools and we give some examples.  
**4.** As babette is designed to be of high quality and extendable, we conclude by describing the further development of the package.

**Keywords:** computational biology, evolution, phylogenetics, BEAST2, R

# Introduction

Phylogenies are commonly used to explore evolutionary hypotheses. Not only can phylogenies show us how species (or other evolutionary units) are related to each other, but we can also estimate relevant parameters such as extinction and speciation rates from them. There are many phylogenetics tools available to construct the phylogeny of a given set of species. BEAST2 () is one of the most widely used ones. It uses a Bayesian statistical framework to estimate the joint posterior distribution of phylogenies and model parameters, from one or more DNA, RNA or amino acid alignments (see figure 1 for an overview of the workflow).

BEAST2 has a graphical and a command-line interface, that both need a configuration file containing alignments and model parameters. BEAST2 is bundled with BEAUti 2 () (’BEAUti’ from now on), a desktop application to create a BEAST2 configuration file. BEAUti has a user-friendly graphical user interface, with helpful default settings. As such, BEAUti is an attractive alternative to manual and error-prone editing of BEAST2 configuration files.

However, BEAUti cannot be called from a command-line script. This implies that when the user wants to explore the consequences of various settings, this must be done manually. This is the manageable workflow when using a few alignments and doing a superficial analysis of sensitivity of the reconstructed tree to model settings. For exploring many trees (for instance from simulations) and for more thorough sensitivity analysis, one would like to loop through multiple (simulated) alignments, nucleotide substitution models, clock models and tree priors. One such tool to replace BEAUti is BEASTmasteR (), which focuses on morphological traits and tip-dating, but also supports DNA data. BEASTmasteR, however, requires hundreds of lines of R code to setup the BEAST2 model configuration and a Microsoft Excel file to specify alignment files.

BEAST2 is also associated with Tracer () and DensiTree (). Both are desktop applications to analyze the output of BEAST2, each with a user-friendly graphical user interface. Tracer’s purpose is to analyze the parameter estimates generated from a BEAST2 run. It shows, among others, the effective sample size (ESS) and time series (’the trace’, hence the name) of each variable in the MCMC run. Both ESS and trace are needed to assess the strength of the inference. DensiTree visualizes the phylogenies of a BEAST2 posterior, with many options to improve the simultaneous display of many phylogenies.

However, for exploring the output of many BEAST2 runs, one would like a script to collect all parameters’ ESSes, parameter traces and posterior phylogenies. There is no single package that offers a complete solution, but examples of R packages that offer a partial solution are rBEAST () and RBeast (). RBeast provides some plotting options and parsing of BEAST2 output files, but the plotting functions are too specific for general use. rBEAST was developed to test a particular biological hypothesis (), and hence was not designed for general use.

Here, we present babette: ’BEAUti 2, BEAST2 and Tracer for R’, which creates BEAST2 (v.2.4.7) configuration files, runs BEAST2, and analyzes its results, all from an R function call. This will save time, tedious mouse clicking and reduces the chances of errors in such repetitive actions. The interface of babette mimics the tools it is based on. This familiarity helps both beginner and experienced BEAST2 users to make the step from those tools to babette. babette enables the creation of a single-script pipeline from sequence alignments to posterior analysis in R.

# Description

babette is written in the R programming language () and enables the full BEAST2 workflow from a single R function call, in a similar way to what subsequent usage of BEAUti, DensiTree and Tracer would produce. babette’s main function is bbt\_run, which configures BEAST2, runs it and parses its output. bbt\_run needs at least the name of a FASTA file containing a DNA alignment. The default settings for the other arguments of bbt\_run are identical to BEAUti’s and BEAST2’s default settings. Per alignment, a site model, clock model and tree prior can be chosen. Multiple alignments can be used, each with its own (unlinked) site model, clock model and tree prior.

babette currently has 108 exported functions to set up a BEAST2 configuration file. babette can currently handle the majority of BEAUti use cases. Because of BEAUti’s high number of plugins, babette uses a software architecture that is designed to be extended. Furthermore, babette has 13 exported functions to run and help run BEAST2. One function is used to run BEAST2, another one installs BEAST2 to a default location. Finally, babette has 21 exported function to parse the BEAST2 output files and analyze the created posterior. babette gives the same ESSes and summary statistics as Tracer. The data is formatted such that it can easily be visualized using ggplot2 (for a trace, similar to Tracer) or phangorn () (for the phylogenies in a posterior, similar to DensiTree).

Currently, babette does not contain all functionality in BEAUti, BEAST2 and their many plug-ins, because these tools themselves also change in time. babette currently works only on DNA data, because this is the most common use case. Nevertheless, babette provides the majority of default tree priors and supports the most important command-line arguments of BEAST2, provides the core Tracer analysis options, and has the most basic subset of plotting options of DensiTree. Up till now, the babette features implemented are those requested by users. Further extension of babette will be based on future user requests.

# Usage

babette can be installed easily from CRAN **NOTE: This is not true yet: we are still in the process of submitting to CRAN**:

install.packages("babette")

For the most up-to-date version, one can download and install the package from babette’s GitHub repository:

devtools::install\_github("richelbilderbeek/babette")

To start using babette, load its functions in the global namespace first:

library(babette)

Because babette calls BEAST2, BEAST2 must be installed. This can be done from R, using:

install\_beast2()

This will install BEAST2 to the default user data folder, but a different path can be specified as well. BEAUti, and likewise babette, needs at least a FASTA filename to produce a BEAST2 configuration file. In BEAUti, this is achieved by loading a FASTA file, then saving an output file using a common save file dialog. After this, BEAST2 needs to be applied to the created configuration file. It creates multiple files storing the posterior. These output files must be parsed by either Tracer or DensiTree. In babette, all this is achieved by:

out <- bbt\_run(fasta\_filenames = "anthus\_aco.fas")

This code will create a (temporary) BEAST2 configuration file, from the FASTA file with name anthus\_aco.fas (which is supplied with the package, from ()), using the same default settings as BEAUti, which are, among others, a Jukes-Cantor site model, a strict clock, and a Yule birth tree prior. babette will then execute BEAST2 using that file, and parses the output. The returned data structure, named out, is a list of parameter estimates (called estimates), posterior phylogenies (called anthus\_aco\_trees, named after the alignment’s name) and MCMC operator performance (operators). An example of using a different site model, clock model and tree prior is:

out <- bbt\_run(  
 fasta\_filenames = "anthus\_aco.fas",  
 site\_models = create\_hky\_site\_model(),  
 clock\_models = create\_rln\_clock\_model(),  
 tree\_priors = create\_bd\_tree\_prior()  
)

This code uses an HKY site model, a relaxed log-normal clock model and a birth-death tree prior, each with their default settings in BEAUti. Table [tab:functions] shows an overview of all functions to create site models, clock models and tree priors. Note that the arguments’ names site\_models, clock\_models and tree\_priors are plural, as each of these can be (a list of) one or more elements. Each of these arguments must have the same number of elements, so that each alignment has its own site model, clock model and tree prior. An example of two alignments, each with its own site model, is:

out <- bbt\_run(  
 fasta\_filenames = c(  
 "anthus\_aco.fas",   
 "anthus\_nd2.fas"  
 ),  
 site\_models = list(  
 create\_tn93\_site\_model(),   
 create\_gtr\_site\_model()  
 )  
)

babette also uses the same default prior distributions as BEAUti for each of the site models, clock models and tree priors. For example, by default, a Yule tree prior assumes that the birth rate follows a uniform distribution, from minus infinity to plus infinity. One may prefer a different ddistribution instead. Here is an example how to specify an exponential distribution for the birth rate in a Yule tree prior in babette:

out <- bbt\_run(  
 fasta\_filenames = "anthus\_aco.fas",  
 tree\_priors = create\_yule\_tree\_prior(  
 birth\_rate\_distr = create\_exp\_distr()   
 )  
)

In this same example, one may specify the initial shape parameters of the exponential distribution. In BEAST2’s implementation, an exponential distribution has one shape parameter: its mean, which can be set to any value with BEAUti. To set the mean value of the exponential distribution to a fixed (non-estimated) value, do:

out <- bbt\_run(  
 fasta\_filenames = "anthus\_aco.fas",  
 tree\_priors = create\_yule\_tree\_prior(  
 birth\_rate\_distr = create\_exp\_distr(  
 mean = create\_mean\_param(  
 value = 1.0,   
 estimate = FALSE  
 )  
 )   
 )  
)

babette also supports node dating. Like BEAUti, one can specify Most Recent Common Ancestor (’MRCA’) priors. An MRCA prior allows to specify taxa having a common ancestor, including a distribution for the date of that ancestor. An application of such a prior is to specify a (close to) fixed crown age, by including all taxa and assuming a tight normal distribution around the desired crown age. With babette, this is achieved as follows:

out <- bbt\_run(  
 fasta\_filenames = "anthus\_aco.fas",  
 mrca\_priors = create\_mrca\_prior(  
 taxa\_names = get\_taxa\_names("anthus\_aco.fas"),  
 alignment\_id = get\_alignment\_id("anthus\_aco.fas"),  
 mrca\_distr = create\_normal\_distr(  
 mean = create\_mean\_param(value = 15.0, estimate = FALSE),  
 sigma = create\_sigma\_param(value = 0.01, estimate = FALSE)  
 )  
 )  
)

Instead of using all taxa, any subset of taxa can be selected to have a dated common ancestor.

babette allows for the same core functionality as Tracer to show the values of the parameter estimates sampled in the BEAST2 run. This is called the “trace” (hence the name). The start of the trace, called the “burn-in”, is usually discarded, as an MCMC algorithm (such as used by BEAST2) first has to converge to its equilibrium and hence the parameter estimates are not representative. By default, Tracer discards the first 10% of all the parameter estimates. To remove a 20% burn-in from all parameter estimates in babette, the following code can be used:

traces <- remove\_burn\_ins(  
 traces = out$estimates,   
 burn\_in\_fraction = 0.2  
)

Tracer shows the ESSes of each posterior’s variables. These ESSes are important to determine the strength of the inference. As a rule of thumb, an ESS of 200 is acceptable for any parameter estimate. To calculate the effective sample sizes (of all estimated variables) in babette:

esses <- calc\_esses(  
 traces = traces,   
 sample\_interval = 1000  
)

Tracer displays multiple summary statistics for each estimated variable: the mean and its standard error, standard deviation, variance, median, mode, geometric mean, 95% highest posterior density interval, auto-correlation time and effective sample size. It displays these statistics per variable. In babette, these summary statistics are collected for all estimated parameters at once:

sum\_stats <- calc\_summary\_stats(  
 traces = traces,   
 sample\_interval = 1000  
)

babette allows for the same functionality as DensiTree. DensiTree displays the phylogenies in a posterior at the same time scale, drawn one over one another, allowing to see the uncertainty in topology and branch lengths. The posterior phylogenies are stored as anthus\_aco\_trees in the object out, and can be plotted as follows:

plot\_densitree(phylos = out$anthus\_aco\_trees)

Instead of running the full pipeline, babette also allows to only create a BEAST2 configuration file. To create a BEAST2 configuration file, with all settings to default, use:

create\_beast2\_input\_file(  
 input\_filenames = babette::get\_babette\_path("anthus\_aco.fas"),  
 output\_filename = "beast2.xml"  
)

This file can then be loaded and edited by BEAUti, run by BEAST2, or run by babette:

run\_beast2(  
 input\_filename = "beast2.xml",  
 output\_log\_filename = "run.log",  
 output\_trees\_filenames = "posterior.trees",  
 output\_state\_filename = "final.xml.state"  
)

In this example, we specify the names of the desired BEAST2 output files. These output files can then be inspected with other tools, or used to continue a BEAST2 run. bbt\_run supports specifying the folder and name of these files, which defaults to a temporary folder to keep the working directory clean of intermediate and temporary files.

# babette resources

babette is free, libre and open source software available at <http://github.com/richelbilderbeek/babette> and is licensed under the GNU General Public License v3.0. babette uses the Travis CI (<https://travis-ci.org>) continuous integration service, which is known to significantly increase the number of bugs exposed () and increases the speed at which new features are added (). babette has a 100% code coverage, which correlates with code quality (). babette follows Hadley Wickham’s style guide (), which improves software quality (). babette depends on multiple packages, which are ape (), beautier (), beastier (), devtools (), geiger (), ggplot2 (), knitr (), phangorn (), rmarkdown (), seqinr (), stringr (), testit () and tracerer (). We tested babette to give a clean error message for incorrect input, by calling babette one million times with random or random sensible inputs, using the Peregrine high performance computer cluster. The test scripts are supplied with babette.

babette’s development takes place on GitHub, <https://github.com/richelbilderbeek/babette>, which accommodates collaboration () and improves transparency (). babette’s GitHub facilitates feature requests and has guidelines how to do so.

babette’s documentation is extensive. All functions are documented in the package’s internal documentation. For quick use, each exported function shows a minimal example. For easy exploration, each exported function’s documentation links to related functions. Additionally, babette has a vignette that demonstrates extensively how to use it. There is documentation on the GitHub to get started, with a dozen examples of BEAUti screenshots with equivalent babette code. Finally, babette has tutorial videos that can be downloaded or viewed on YouTube, <https://goo.gl/weKaaU>.

# Citation of babette

Scientists using babette in a published paper can cite this article, and/or cite the babette package directly. To obtain this citation from within an R script, use:

> citation("babette")

# Acknowledgements

Thanks to Yacine Ben Chehida and Paul van Els for supplying their BEAST2 use cases. Thanks again to Paul van Els for sharing his FASTA files for use by this package. Thanks to Leonel Herrera-Alsina, Raphael Scherrer and Giovanni Laudanno for their comments on this package and article. Thanks to Huw Ogilvie and one anonymous reviewer for reviewing this article. Thanks to rOpenSci, and especially Noam Ross and Guangchuang Yu for reviewing the package’s source code. We would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster. We thank the Netherlands Organization for Scientific Research (NWO) for financial support through a VICI grant awarded to RSE.

# Authors’ contributions

RJCB and RSE conceived the idea for the package. RJCB created and tested the package, and wrote the first draft of the manuscript. RSE contributed substantially to revisions.

babette’s main functions

|  |  |
| --- | --- |
| **Name** | **Description** |
| bbt\_run | Run BEAST2 |
| create\_gtr\_site\_model | Create a GTR site model |
| create\_hky\_site\_model | Create an HKY site model |
| create\_jc69\_site\_model | Create a Jukes-Cantor site model |
| create\_tn93\_site\_model | Create a TN93 site model |
| create\_rln\_clock\_model | Create a relaxed log-normal clock model |
| create\_strict\_clock\_model | Create a strict clock model |
| create\_bd\_tree\_prior | Create a birth-death tree prior |
| create\_cbs\_tree\_prior | Create a coalescent Bayesian skyline tree prior |
| create\_ccp\_tree\_prior | Create a coalescent constant-population tree prior |
| create\_cep\_tree\_prior | Create a coalescent exponential-population tree prior |
| create\_yule\_tree\_prior | Create a Yule tree prior |
| create\_beta\_distr | Create a beta distribution |
| create\_exp\_distr | Create an exponential distribution |
| create\_gamma\_distr | Create a gamma distribution |
| create\_inv\_gamma\_distr | Create an inverse gamma distribution |
| create\_laplace\_distr | Create a Laplace distribution |
| create\_log\_normal\_distr | Create a log-normal distribution |
| create\_normal\_distr | Create a normal distribution |
| create\_one\_div\_x\_distr | Create a 1/X distribution |
| create\_poisson\_distr | Create a Poisson distribution |
| create\_uniform\_distr | Create a uniform distribution |