rbeast2: 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, rbeast2: ’BEAUti 2, BEAST2 and Tracer for R’, for the R programming language. rbeast2 creates BEAST2 input files, runs BEAST2 and parses its results, all from an R function call.  
**3.** We describe rbeast2’s usage and the novel functionality it provides compared to the original tools and give some examples.  
**4.** As rbeast2 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) relate to each other, but we also estimate relevant parameters such as extinction and speciation rates.

There are many phylogenetics tools available to obtain an estimate of the phylogenetic tree of a given set of species. BEAST2 is one of the most widely used ones. It creates a posterior of jointly-estimated phylogenies and model parameters, from one or more DNA, RNA or amino acid alignments (see figure [fig:workflow] for an overview of the workflow). It is a console application, that needs 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 and reasonable 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 common 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. Creating BEAUti configuration files in BEASTmasteR is however quite cumbersome.

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 BEAST 2 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 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. Examples of R packages to do some of these tasks are rBEAST and RBeast.

Here, we present rbeast2: ’BEAUti 2, BEAST2 and Tracer for R’, which creates BEAST2 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 rbeast2 mimics the tools it is based on. This familiarity helps both beginner and experienced BEAST2 users to make the step from those tools to rbeast2. rbeast2 enables the creation of a single-script pipeline from sequence alignments to posterior analysis in R.

rbeast2 is the first R package that unifies the full workflow of working with BEAST2. Whilst BEASTmasteR needs dozens for lines to create a BEAST2 configuration file, with rbeast2 the same can be created by a simple one-liner. Unlike rBEAST and RBeast, rbeast2 implements all of Tracer’s primary functionality. This makes rbeast2 a complete solution for using BEAST2 from R.

# Description

rbeast2 is written in the R programming language and enables the full BEAST2 workflow from an R function call, in a similar way to what BEAUti, DensiTree and Tracer do.

rbeast2’s main function is run\_beast2, which configures BEAST2, runs it and parses its output. run\_beast2 needs at least the name of a FASTA file containing a DNA alignment. The default settings for the other arguments of run\_beast2 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.

rbeast2 currently has 61 exported functions to set up a BEAST2 configuration file. rbeast2 is an alternative for a majority of BEAUti use cases, but does not yet support the full functionality of BEAUti. Because of BEAUti’s high number of plugins, rbeast2 uses a software architecture that expects to be extended.

Furthermore, rbeast2 has 7 exported function to run and help run BEAST2. One function is used to run BEAST2, others allow the user to check if a BEAST2 configuration file is indeed valid.

Finally, rbeast2 has 20 exported function to parse the BEAST2 output files and analyze the created posterior. rbeast2 gives the same ESSes and summary statistics as Tracer. The data is formatted as 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, rbeast2 does not replace all functionality in BEAUti, as it does not provide 3 out of 7 tree priors, nor does it support RNA alignments or use of morphological data. The many plug-ins of BEAUti are not yet supported by rbeast2. rbeast2 does not support all command-line arguments of BEAST2, does not provide the more specialized Tracer analysis options, nor is it as feature-rich in plotting options as DensiTree. Up until now, the rbeast2 features implemented are those requested by users. Further extensions of rbeast2 will be based on future user requests.

# Examples

In R, the functions of a package need to be loaded in the global namespace first:

library(rbeast2)

BEAUti, and likewise rbeast2, 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 of DensiTree.

In rbeast2, all this is achieved by:

out <- run\_beast2("alignment.fas")

This code will create a (temporary) BEAST2 configuration file, from the FASTA file with name alignment.fas, 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. rbeast2 will then execute BEAST2 using that file, and parses the output. The returned data structure, named out, is a list of parameter estimates, posterior phylogenies (one for each alignment) and MCMC operator performance.

An example of using a different site model, clock model and tree prior is:

out <- run\_beast2(  
 "alignment.fas",  
 site\_models = hky\_site\_model(),  
 clock\_models = rln\_clock\_model(),  
 tree\_priors = 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.

rbeast2’s main functions

|  |  |
| --- | --- |
| **Name** | **Description** |
| run\_beast2 | 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 |

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 <- run\_beast2(  
 c("anthus\_aco.fas", "anthus\_nd2.fas"),  
 site\_models = list(  
 create\_tn93\_site\_model(),   
 create\_gtr\_site\_model()  
 )  
)

rbeast2 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. This assumption implies that negative and positive birth rates are just as likely, where a negative birth rate is biologically impossible (note that in practice, this usually works out just fine). One may prefer an exponential distribution instead, as this would assume only positive birth rates, and makes high birth rates unlikely.

To do this in rbeast2:

out <- run\_beast2(  
 "alignment.fas",  
 tree\_priors = create\_yule\_tree\_prior(  
 birth\_rate\_distr = create\_exp\_distr()   
 )  
)

Our initial motivation to create rbeast2 was that we wanted to fix the crown age of a phylogeny. BEAUti assumes that a phylogeny has a crown age that needs to be jointly-estimated with the phylogeny and other parameters. It does not allow for fixing the crown age. Without rbeast2, one needs to manually edit the BEAST2 configuration file, which is tedious and prone to errors. Fixing the crown ages is especially useful for theoretical experiments, as this allows for one less source of variation.

This is how to specify a fixed crown age with rbeast2:

out <- run\_beast2(  
 "alignment.fas",  
 posterior\_crown\_age = 15  
)

rbeast2 allows for the same functionality as Tracer. Tracer works on the values of the parameter estimates sampled in the BEAST2 run. This is called the “trace” (hence the name). The start of the trace is usually discarded, as an MCMC algorithm (such as used by BEAST2) first has to converge to its equilibrium. The start of the trace, called the “burn-in”, will be removed, as its parameter estimates are not representative. By default, Tracer discards the first 10% of all the parameter estimates.

To remove the burn-in from all parameter estimates in rbeast2, the following code can be used:

traces <- remove\_burn\_ins(out$estimates)

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 rbeast2:

esses <- calc\_esses(  
 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 rbeast2, these summary statistics are collected for all estimated parameters at once:

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

rbeast2 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. To do so in rbeast2:

densitree(out$anthus\_aco\_trees)

# rbeast2 resources

rbeast2 is free, libre and open source software available from the official R package archive at <http://cran.r-project.org/src/contrib/PACKAGES.html#rbeast2> and is licensed under the GNU General Public License v3.0.

rbeast2 uses the Travis CI continuous integration service, which is known to significantly increase the number of bugs exposed and increases the speed at which new features are added. rbeast2 has a 100% code coverage, which correlates with code qualityand it follows Hadley Wickham’s style guide , which improves software quality rbeast2 depends on multiple packages, which are ape , beautier , beastier , devtools , geiger , ggplot2 , knitr , phangorn , rmarkdown , seqinr , stringr , testit and tracerer .

rbeast2’s development takes place on GitHub , <https://github.com/richelbilderbeek/rbeast2>, which accommodates collaboration and improves transparency. rbeast2’s GitHub facilitates feature requests and has guidelines how to do so.

rbeast2’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, rbeast2 has a vignette that demonstrates extensively how to use it. The GitHub documentation helps the user to get started, with a dozen examples of BEAUti screenshots with equivalent rbeast2 code.

# Citation of rbeast2

Scientists using rbeast2 in a published paper can cite this article, and/or cite the rbeast2 package directly. To obtain this citation from within an R script, use:

> citation("rbeast2")