beautier: BEAUti for R

Richèl J.C. Bilderbeek, Rampal S. Etienne

1. Here, we present a package, `beautier`, 'BEAUti for R', for the R programming language.
2. `beautier` allows for scripted use of the BEAST2 phylogenetics tool, by creating BEAST2 input files from an R function call.
3. I describe `beautier` usage, the novel functionality it provides compared to BEAUti, and give some minimal examples.
4. As `beautier` is free, libre, open-source and designed to be extended, I conclude by describing the current development of the package

Introduction

I would say it is better to start out by mentioning (maybe briefly) what phylogenies are used for and that there is an increase in the application of them for different purposes. Maybe introduce the reader to the field of systematics.

Con formato: Texto independiente

BEAST2 is a Bayesian phylogenetics tool. BEAST2 creates a posterior of jointly estimated phylonies and model parameters, from a DNA, RNA or amino acid alignment. BEAST2 can be run from the command line. To run BEAST2 from the command-line, an XML configuration file is needed, that contains the alignment and model parameters.

BEAST2 is bundled with BEAUti . BEAUti is a program to create a BEAST2 XML configuration file, with a user-friendly graphical user interface. BEAUti's purpose is to help new BEAST2 users getting started. To do so, BEAUti starts with reasonable default settings. To create a minimal BEAST2 configuration file, BEAUti only needs one alignment file. BEAUti cannot be used from within shell scripts.

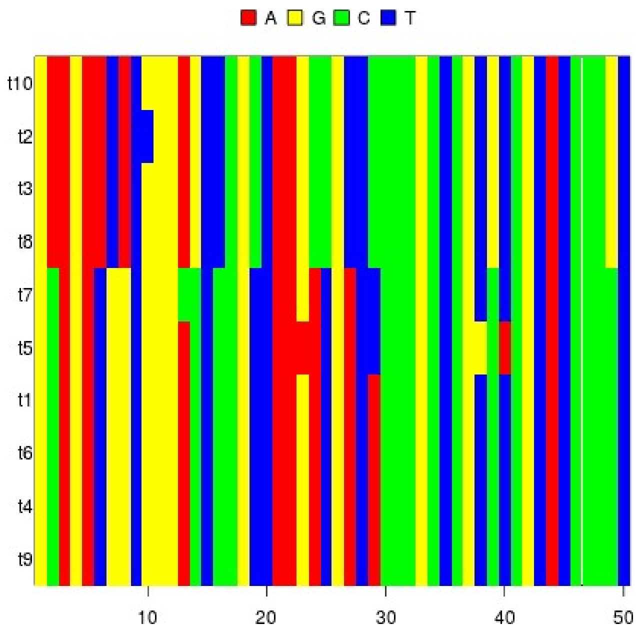
`beautier`, 'BEAUti for R', allows for creating BEAST2 configuration files from an R function call. The interface of `beautier` mimics BEAUti. This familiar interface helps experienced BEAST2 users to create configuration' files from shell scripts.

Comentado [T1]: In here, you are saying what your paper is about. I would suggest to start with the typical: "Here, we present…."

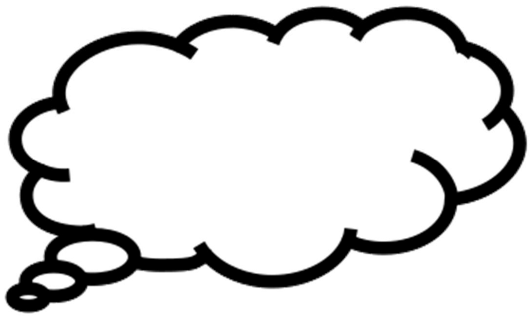
BEAUti does not allow yet that an inferred phylogeny's crown age is fixed. In an empirical context, a phylogeny's crown age is one of the estimated parameters. For theoretical work, a fixed and known crown age can result in a cleaner analysis. `beautier` allows to specify a fixed crown age.

Comentado [T2]: I think this does not belong to introduction, place move it elsewhere (maybe discussion?)

Remember that figures go to the end of the ms, each one in a separate page. All the figure captions in a single page, at the end of the ms too.



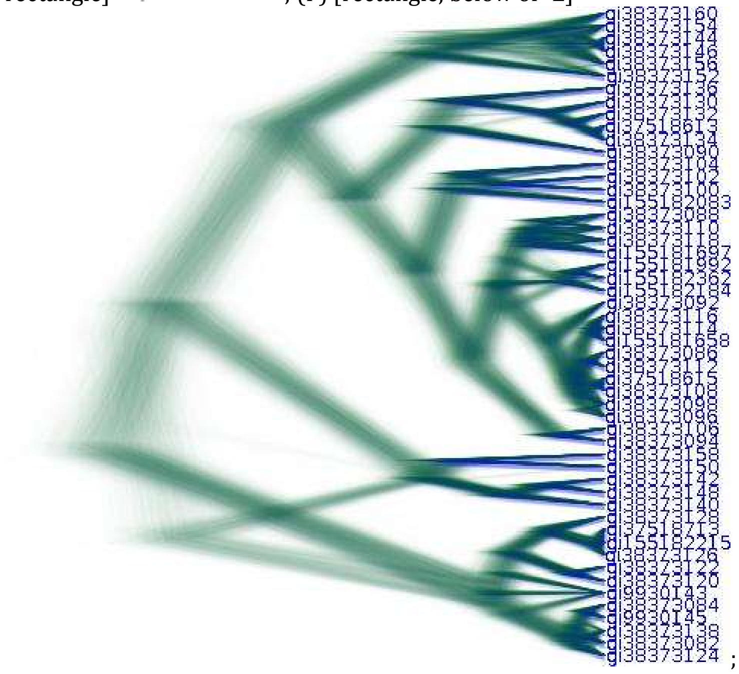
=[] (A) [rectangle]



; (E) [below of=A,

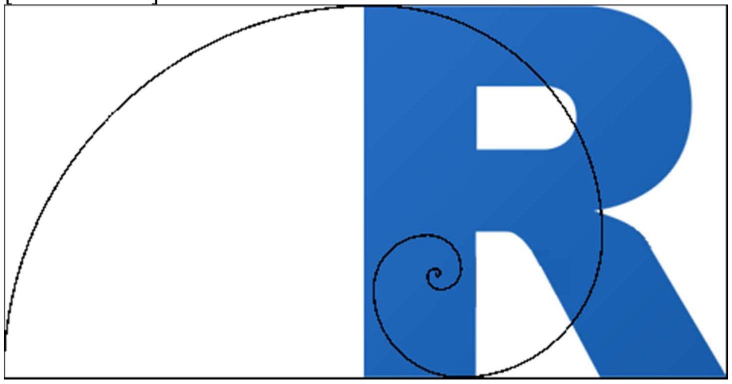
```
<?xml>
<beast>
  ...
```

rectangle] `</beast>` ; (F) [rectangle, below of=E]



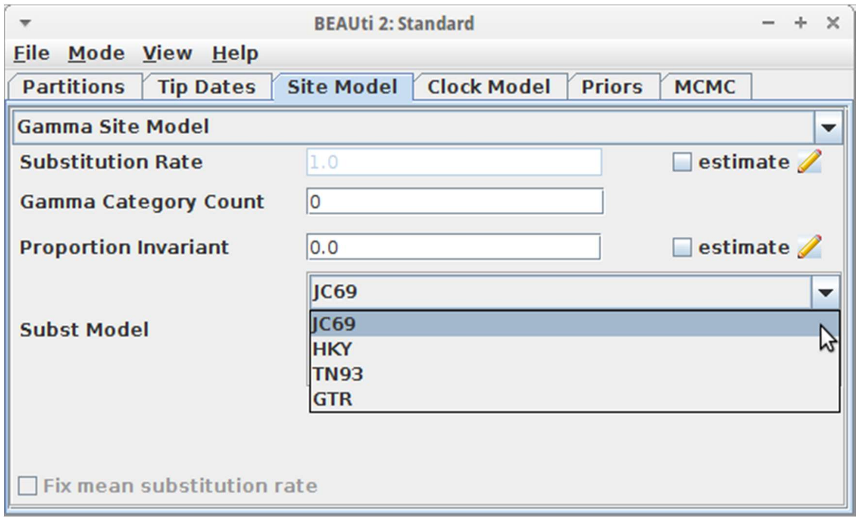
; (A) edge

[anchor = east] node



(E) (A) edge

[anchor = west] node


BEAUti 2: Standard

File Mode View Help

Partitions | Tip Dates | Site Model | Clock Model | Priors | MCMC

Gamma Site Model

Substitution Rate: 1.0 [ ] estimate

Gamma Category Count: 0

Proportion Invariant: 0.0 [ ] estimate

Subst Model: JC69 (JC69, HKY, TN93, GTR)

[ ] Fix mean substitution rate

(E)

(E) edge [] node (F);

Descriptions

`beautier` is written in the R programming language . `beautier` creates the BEAST2 input files from an R function call, in a similar way that BEAUti does.

`beautier`'s main function is `create_beast2_input_file`, which creates an BEAST2 input file. `create_beast2_input_file` needs at least the name of a FASTA file containing a DNA alignment and a name for the to-be-created output file. This interface follows BEAUti's default settings. Per alignment, a site model, clock model and tree priors can be chosen. Multiple alignments can be used, each with its own (unlinked) site model, clock model and tree prior.

beautier's functions

Name	Description
`create_beast2_input_file`	Creates a BEAST2 input file
`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

In total, `beautier` has 59 exported functions to create a BEAST2 configuration file. `beautier` is an alternative for a majority of BEAUti use cases. `beautier` does not support the full functionality of BEAUti. Considering the size, age and number of plugins, this would be close to impossible. To compensate for this, an extensible software architecture is used. `beautiers` future extensions can be found on its GitHub.

BEAUti assumes that a phylogeny has a crown age that needs to be jointly estimated with the phylogeny and other parameters. BEAUti does not allow for fixing a phylogeny's crown age. Before `beautier`, one needs to manually edit the BEAST2 XML configuration file, which is prone to errors. `beautier`, allow easy fixing of phylogenies' crown ages.

Comentado [T3]: This paragraph is similar to something you wrote earlier

`beautier` has only internal support for calling BEAST2 from within R and does so for testing purposes only. `beautier` has minimal support for parsing and interpreting BEAST2 output files, for that the `beastier` package is recommended.

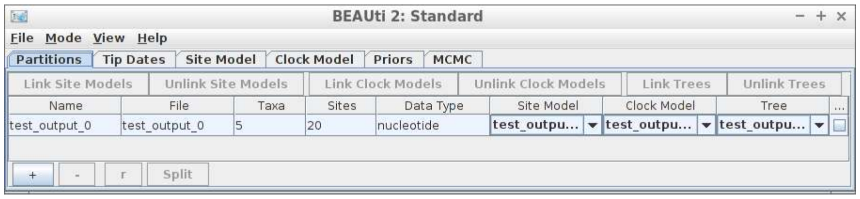
Examples

In R, a package's function needs to be loaded in the global namespace first:

```
library(beautier)
```

BEAUti, and likewise `beautier`, need at least a FASTA filename and an XML output filename. In BEAUti, this is achieved by loading a FASTA file (resulting in figure [fig:simplest_beauti_usage]), then saving an output file using a common save file dialog. In `beautier`, the same is achieved by listing [lst:simplest_example]:

```
library(beautier)
create_beast2_input_file(
  "alignment.fas",
  "beast2.xml"
)
```


BEAUti 2: Standard
File Mode View Help
Partitions Tip Dates Site Model Clock Model Priors MCMC
Link Site Models Unlink Site Models Link Clock Models Unlink Clock Models Link Trees Unlink Trees
+ - r Split

Name	File	Taxa	Sites	Data Type	Site Model	Clock Model	Tree
test_output_0	test_output_0	5	20	nucleotide	test_outpu...	test_outpu...	test_outpu...

Simplest BEAUti usage

This code will create a BEAST2 file with name '`beast2.xml`', using a FASTA file with name `alignment.fas`, using the same default settings as BEAUti. The default settings are, among others, to use a Jukes-Cantor site model , a strict clock, and a Yule birth tree prior . Listing [lis:simplest_example_explicit] shows how to explicitly pick these settings:

```
library(beautier)
create_beast2_input_file(
  input_fasta_filenames = "alignment.fas",
  output_xml_filename = "beast2.xml",
  site_models = create_jc69_site_model(),
  clock_models = create_strict_clock_model(),
  tree_priors = create_yule_tree_prior()
)
```

The argument names `input_fasta_filenames`, `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 using a different site model, clock model and tree prior is shown by listing [lis:all_different]:

```
library(beautier)
create_beast2_input_file(
  input_fasta_filenames = "alignment.fas",
  output_xml_filename = "beast2.xml",
  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 . Table [tab:functions] shows an overview of all functions to create site models, clock models and tree priors.

`beautier` creates site models, clock models and tree priors with the same default distributions as BEAUti. For example, a Yule tree prior assumes that birth rate likelihoods follow a uniform distribution, from minus infinity to infinity. This assumption entails that negative and positive birth rates are just as likely, where a negative birth rate is biologically impossible. One may prefer to have an exponential distribution instead, as this would state that birth rates are always positive, and higher values are less likely than lower values. To do so `beautier` is shown by listing [lis:diff_distr]:

```
library(beautier)
create_beast2_input_file(
  input_fasta_filenames = "alignment.fas",
  output_xml_filename = "beast2.xml",
  tree_priors = create_yule_tree_prior(
    birth_rate_distr = create_exp_distr()
  )
)
```

Novel about `beautier` is that it allows for specifying a fixed crown age. By default, a phylogeny's crown age is jointly estimated with the other parameters. Setting a fixed crown age is not yet possible in BEAUti directly, but it is documented how to manually edit the XML file to allow for a fixed crown age. Listing [lst:fixed_crown_age] shows how to specify a fixed crown age is `beautier`:

```
create_beast2_input_file(
  "alignment.fas",
  "beast2.xml"
  fixed_crown_ages = TRUE,
  initial_phylogenies = fasta_to_phylo(
    fasta_filename = "alignment.fas",
    crown_age = 15
  )
)
```

This code shows that setting `fixed_crown_age` to true is insufficient. An initial phylogeny of the desired (fixed) crown age needs to be supplied. In this example, a random phylogeny is constructed from the FASTA filename. Supplying an informed phylogeny will make the MCMC algorithm start at a likelier initial state.

beautier development and other resources

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

`beautier`'s development takes place on GitHub . GitHub is a website that hosts (among others) software and facilitates its development. Using GitHub is a good practice for computational scientists and improves transparency .

Comentado [T4]: Maybe this information is not tightly linked to your ms

`beautier`'s quality is assured by Travis CI . Travis CI is a continuous integration service, that runs a script upon a (suggested) change in `beautier`'s code. The `beautier` script checks if the package can be build, runs all unit and integration tests, measures code coverage, coding style and good practices.

Unit and integration tests check the integrity of `beatier`. All functions are tested for producing the correct output and desired error handling. Travis CI uses the testing facilities of the `testthat` package.

Code coverage is the percentage of code that is executed in tests. Code coverage correlates with code quality . Travis CI uses the `covr` package to measure code coverage. `beautier` has a 100% code coverage.

Coding style is the way statements are laid out, for example the placement of curly brackets. `beautier` follows Hadley Wickham's style guide . Travis CI uses the `lintr` package to confirm this coding style is used.

Good practices are miscellaneous things considered good practices. An example of a good practice is (next to high code coverage and consistent coding style) to have short functions with a low cyclomatic complexity. Travis CI uses the `goodpractice` package to comfirm all good practices are followed.

Comentado [T5]: The same is true for all of this

`beautier` is dependent on multiple packages, which are `APE` , `devtools` , `geiger` , `ggplot2` , `knitr` , `phangorn` , `beastier` , `rmarkdown` , `seqinr` , `stringr` , `testit` and `TreeSim` .

`beautier`'s documentation is extensive, yet concise. 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, `beautier` has a vignette that demonstrates in a longer form how to use it. The integrity of this documentation is tested each time the package is built by Travis CI. The documentation on the GitHub helps to get started, with a dozen examples of a BEAUti screenshot and the equivalent `beautier` code.

`beautier`'s GitHub facilitates feature requests and guidelines how to do so. New code can be submitted using GitHub's infrastructure (a 'Pull Request'). New code will be tested by Travis CI to follow the same quality standards and only accepted when there are no

Citation of beautier

Scientists using `beautier` in a published paper should cite this article. Users can additionally cite the `beautier` package directly. Citation information can be obtained by typing:

```
> citation("beautier")
```

from within R.