

# Package ‘beastier’

August 11, 2022

**Type** Package

**Title** Call 'BEAST2'

**Version** 2.4.11

**Maintainer** Richèl J.C. Bilderbeek <[richel@richelbilderbeek.nl](mailto:richel@richelbilderbeek.nl)>

**Description** 'BEAST2' (<<https://www.beast2.org>>) is a widely used Bayesian phylogenetic tool, that uses DNA/RNA/protein data and many model priors to create a posterior of jointly estimated phylogenies and parameters.

'BEAST2' is a command-line tool.

This package provides a way to call 'BEAST2' from an 'R' function call.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Imports** ape, assertive, beautier (>= 2.6.5), devtools, phangorn, rappdirs, readr, rJava, stringr, tibble, xml2

**Suggests** hunspell, knitr, markdown, rmarkdown, spelling, testit, testthat (>= 2.1.0), tracerer

**URL** <https://docs.ropensci.org/beastier/> (website)

<https://github.com/ropensci/beastier/>

**BugReports** <https://github.com/ropensci/beastier>

**Language** en-US

**VignetteBuilder** knitr

**SystemRequirements** BEAST2 (<https://www.beast2.org/>)

**NeedsCompilation** no

**Author** Richèl J.C. Bilderbeek [aut, cre]

(<<https://orcid.org/0000-0003-1107-7049>>),

Joëlle Barido-Sottani [rev] (Joëlle reviewed the package for rOpenSci, see <https://github.com/ropensci/onboarding/issues/209>),

David Winter [rev] (David reviewed the package for rOpenSci, see

<https://github.com/ropensci/onboarding/issues/209>),  
 Jason Griffiths [ctb] (<<https://orcid.org/0000-0002-1667-8233>>),  
 Thijs Janzen [ctb]

**Repository** CRAN

**Date/Publication** 2022-08-11 14:40:04 UTC

## R topics documented:

add_quotes_if_has_spaces . . . . .	4
are_beast2_input_lines . . . . .	5
are_beast2_input_lines_deep . . . . .	6
are_beast2_input_lines_fast . . . . .	7
are_identical_alignments . . . . .	8
beast2_options_to_table . . . . .	8
beastier . . . . .	9
beastier_report . . . . .	10
check_beast2 . . . . .	11
check_beast2_options . . . . .	11
check_beast2_optionses . . . . .	12
check_beast2_options_data_types . . . . .	13
check_beast2_options_do_not_overwrite_existing_files . . . . .	14
check_beast2_options_filenames_differ . . . . .	15
check_beast2_options_names . . . . .	15
check_beast2_path . . . . .	16
check_can_create_dir_for_state_output_file . . . . .	17
check_can_create_file . . . . .	17
check_can_create_screenlog_file . . . . .	18
check_can_create_state_output_file . . . . .	19
check_can_create_tracelog_file . . . . .	19
check_can_create_treelog_file . . . . .	20
check_empty_beastier_folder . . . . .	20
check_empty_beastier_folders . . . . .	21
check_input_filename . . . . .	22
check_input_filename_validity . . . . .	22
check_n_threads . . . . .	23
check_os . . . . .	24
check_rng_seed . . . . .	25
continue_beast2 . . . . .	25
create_beast2_continue_cmd_from_options . . . . .	26
create_beast2_input_file_folder . . . . .	27
create_beast2_options . . . . .	28
create_beast2_run_cmd . . . . .	29
create_beast2_run_cmd_from_options . . . . .	30
create_beast2_screenlog_folder . . . . .	31
create_beast2_state_output_file_folder . . . . .	32
create_beast2_tracelog_folder . . . . .	32
create_beast2_treelog_folder . . . . .	33

create_beast2_validate_cmd . . . . .	33
create_beast2_validate_cmd_bin . . . . .	34
create_beast2_validate_cmd_jar . . . . .	35
create_beast2_version_cmd . . . . .	36
create_beast2_version_cmd_bin . . . . .	37
create_beast2_version_cmd_jar . . . . .	38
create_beastier_tempfolder . . . . .	38
create_mcbette_beast2_options . . . . .	39
create_random_alignment . . . . .	41
create_random_fasta . . . . .	42
create_random_phylogeny . . . . .	43
create_temp_input_filename . . . . .	43
create_temp_state_filename . . . . .	44
default_params_doc . . . . .	45
do_minimal_run . . . . .	48
extract_screenlog_filename_from_beast2_input_file . . . . .	49
extract_tracelog_filename_from_beast2_input_file . . . . .	50
extract_treelog_filename_from_beast2_input_file . . . . .	51
get_alignment_ids_from_xml_filename . . . . .	52
get_beast2_example_filename . . . . .	52
get_beast2_example_filenames . . . . .	53
get_beast2_main_class_name . . . . .	54
get_beast2_options_filenames . . . . .	55
get_beast2_version . . . . .	55
get_beastier_folder . . . . .	56
get_beastier_path . . . . .	57
get_beastier_paths . . . . .	57
get_beastier_tempfilename . . . . .	58
get_default_beast2_bin_path . . . . .	59
get_default_beast2_download_url . . . . .	60
get_default_beast2_download_url_linux . . . . .	61
get_default_beast2_download_url_win . . . . .	61
get_default_beast2_folder . . . . .	62
get_default_beast2_jar_path . . . . .	63
get_default_beast2_path . . . . .	64
get_default_beast2_version . . . . .	65
get_default_java_path . . . . .	65
get_duplicate_param_ids . . . . .	66
get_java_version . . . . .	67
get_trees_filenames . . . . .	67
gives_beast2_warning . . . . .	68
has_unique_ids . . . . .	69
install_beast2 . . . . .	70
is_alignment . . . . .	71
is_beast2_input_file . . . . .	71
is_beast2_installed . . . . .	73
is_bin_path . . . . .	73
is_jar_path . . . . .	74

is_on_appveyor . . . . .	75
is_on_ci . . . . .	76
is_on_travis . . . . .	76
is_win_bin_path . . . . .	77
print_beast2_options . . . . .	77
remove_beastier_folder . . . . .	78
remove_beastier_folders . . . . .	79
remove_file_if_present . . . . .	79
rename_beast2_options_filenames . . . . .	80
run_beast2 . . . . .	81
run_beast2_from_options . . . . .	83
save_lines . . . . .	84
save_nexus_as_fasta . . . . .	84
uninstall_beast2 . . . . .	85
upgrade_beast2 . . . . .	86

<b>Index</b>	<b>87</b>
--------------	-----------

---

add\_quotes\_if\_has\_spaces

*Add quotes around the string if it contains spaces.*

---

### Description

Add quotes around the string if it contains spaces. Does nothing if the string contains no spaces. This is used for filenames

### Usage

```
add_quotes_if_has_spaces(filename)
```

### Arguments

filename          a filename

### Value

a filename. If the filename did not contain spaces, it is returned as-is. If the filename did contain spaces, the filename is surrounded by quotes

### Author(s)

Richèl J.C. Bilderbeek

**Examples**

```

remove_beastier_folders()
check_empty_beastier_folders()

add_quotes_if_has_spaces("x")
add_quotes_if_has_spaces("a b")

check_empty_beastier_folders()

```

---

```
are_beast2_input_lines
```

*Would these lines of text, when written to a file, result in a valid BEAST2 input file?*

---

**Description**

Would these lines of text, when written to a file, result in a valid BEAST2 input file?

**Usage**

```

are_beast2_input_lines(
  lines,
  verbose = FALSE,
  method = ifelse(beastier::is_on_ci(), "deep", "fast"),
  beast2_path = get_default_beast2_path()
)

```

**Arguments**

lines	lines of text
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging
method	the method to check. Can be 'deep' or 'fast'. The 'deep' method uses BEAST2 to validate the complete file. The 'fast' method uses some superficial tests (for example: if all IDs are unique)
beast2_path	name of either a BEAST2 binary file (usually simply beast) or a BEAST2 jar file (usually has a .jar extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path

**Value**

TRUE if the text is valid, FALSE if not

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [is\\_beast2\\_input\\_file](#) to check a file

**Examples**

```
if (is_beast2_installed() && beautier::is_on_ci()) {
  are_beast2_input_lines(get_beastier_path("anthus_2_4.xml"))

  remove_beastier_folders()
}
check_empty_beastier_folders()
```

---

```
are_beast2_input_lines_deep
```

*Would these lines of text, when written to a file, result in a valid BEAST2 input file?*

---

**Description**

Would these lines of text, when written to a file, result in a valid BEAST2 input file?

**Usage**

```
are_beast2_input_lines_deep(
  lines,
  verbose = FALSE,
  beast2_path = get_default_beast2_path()
)
```

**Arguments**

lines	lines of text
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging
beast2_path	name of either a BEAST2 binary file (usually simply beast) or a BEAST2 jar file (usually has a .jar extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path

**Value**

TRUE if the text is valid, FALSE if not

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [is\\_beast2\\_input\\_file](#) to check a file

**Examples**

```
if (is_beast2_installed() && beautier::is_on_ci()) {
  beast2_filename <- get_beastier_path("anthus_2_4.xml")
  text <- readLines(beast2_filename)
  are_beast2_input_lines_deep(text)

  remove_beastier_folders()
}
check_empty_beastier_folders()
```

---

are\_beast2\_input\_lines\_fast

*Would these lines of text, when written to a file, result in a valid BEAST2 input file?*

---

**Description**

Would these lines of text, when written to a file, result in a valid BEAST2 input file?

**Usage**

```
are_beast2_input_lines_fast(lines)
```

**Arguments**

lines            lines of text

**Value**

TRUE if the text is valid, FALSE if not

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [is\\_beast2\\_input\\_file](#) to check a file

## Examples

```
beast2_filename <- get_beastier_path("anthus_2_4.xml")
text <- readLines(beast2_filename)

# TRUE
are_beast2_input_lines_fast(text)
check_empty_beastier_folders()
```

---

are\_identical\_alignments

*Determines if the two alignments are equal*

---

## Description

Determines if the two alignments are equal

## Usage

```
are_identical_alignments(p, q)
```

## Arguments

p	the first alignment
q	the second alignment

## Value

TRUE or FALSE

## Author(s)

Richèl J.C. Bilderbeek

---

beast2\_options\_to\_table

*Convert a beast2\_options to a table*

---

## Description

Convert a beast2\_options to a table

## Usage

```
beast2_options_to_table(beast2_options)
```



## Arguments

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

## Value

a [tibble](#) with two columns, called 'parameter' and 'value'. Each 'parameter' is the name of the element of the 'beast2\_options' structure, where the 'value' on the same row holds the value of that parameter

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
beast2_options_to_table(create_beast2_options())
```

---

beastier	<i>beastier: A package to call BEAST2.</i>
----------	--

---

## Description

`beastier` allows to call BEAST2, a popular Bayesian phylogenetics tool, using an R interface. 'beastier' closely follows the interface of BEAST2, including its default settings.

## See Also

These are packages associated with `beastier`:

- The package `beautier` can create BEAST2 input files from R
- The package `tracerer` can parse BEAST2 output files from R
- The package `babette` combines the functionality of `beautier`, `beastier` and `tracerer` into a single workflow

## Examples

```
check_empty_beastier_folders()

beast2_options <- create_beast2_options(
  input_filename = get_beastier_path("2_4.xml")
)

if (is_beast2_installed() && beautier::is_on_ci()) {
  run_beast2_from_options(beast2_options)
  file.remove(beast2_options$output_state_filename)
  remove_beastier_folders()
}
```

---

beastier_report	<i>Creates a <a href="#">beastier</a> report</i>
-----------------	--

---

### Description

Creates a [beastier](#) report, to be used when reporting bugs. Uses [message](#)

### Usage

```
beastier_report(  
    beast2_folder = get_default_beast2_folder(),  
    os = rappdirs::app_dir()$os  
)
```

### Arguments

beast2_folder	the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a sub-folder. Use <a href="#">get_default_beast2_folder</a> to get the default BEAST2 folder. Use <a href="#">get_default_beast2_bin_path</a> to get the full path to the default BEAST2 executable.
os	name of the operating system, must be unix (Linux, Mac) or win (Windows)

### Value

No return value, the information will be shown using [message](#)

### Author(s)

Richèl J.C. Bilderbeek

### Examples

```
check_empty_beastier_folders()  
  
beastier_report()  
  
check_empty_beastier_folders()
```

---

check_beast2	<i>Check if BEAST2 is installed properly.</i>
--------------	---

---

**Description**

Calls [stop](#) if BEAST2 is improperly installed

**Usage**

```
check_beast2(beast2_path = beastier::get_default_beast2_path())
```

**Arguments**

`beast2_path` name of either a BEAST2 binary file (usually simply `beast`) or a BEAST2 jar file (usually has a `.jar` extension). Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default BEAST binary file's path Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default BEAST jar file's path

**Value**

nothing Will [stop](#) if BEAST2 is improperly installed

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed()) {  
  check_beast2()  
}
```

---

check_beast2_options	<i>Check if the <code>beast2_options</code> is a valid BEAST2 options object.</i>
----------------------	---

---

**Description**

Calls `stop` if the BEAST2 option object is invalid

**Usage**

```
check_beast2_options(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

nothing Will stop if the BEAST2 option object is invalid

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [create\\_beast2\\_options](#) to create a valid BEAST2 options object

**Examples**

```
check_empty_beastier_folders()
```

```
check_beast2_options(create_beast2_options())
```

```
check_empty_beastier_folders()
```

---

check\_beast2\_optionses

*Check if the beast2\_options is a valid BEAST2 options object.*

---

**Description**

Calls stop if the BEAST2 option object is invalid

**Usage**

```
check_beast2_optionses(beast2_optionses)
```

**Arguments**

beast2\_optionses

list of one or more beast2\_options structures, as can be created by [create\\_beast2\\_options](#).  
Use of reduplicated plural to achieve difference with beast2\_options

**Value**

Nothing. Will stop if the BEAST2 option object is invalid

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [create\\_beast2\\_options](#) to create a valid BEAST2 options object

### Examples

```
check_empty_beastier_folders()

check_beast2_optionses(list(create_beast2_options()))

check_empty_beastier_folders()
```

---

check\_beast2\_options\_data\_types

*Check if the `beast2_options`, which is a list, has all elements of the right data types*

---

### Description

Calls `stop` if not.

### Usage

```
check_beast2_options_data_types(beast2_options)
```

### Arguments

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

### Value

nothing

### Author(s)

Richèl J.C. Bilderbeek

### See Also

Use [check\\_beast2\\_options](#) to check the entire `beast2_options` object

---

`check_beast2_options_do_not_overwrite_existing_files`*Internal function*

---

**Description**

Check if the `beast2_options` will not overwrite existing files, when the `'overwrite'` options is set to `FALSE`.

**Usage**

```
check_beast2_options_do_not_overwrite_existing_files(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Details**

Will [stop](#) if a file is threatened to be overwritten

**Value**

Nothing. Will [stop](#) if a file is threatened to be overwritten

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

check_beast2_options_do_not_overwrite_existing_files(
  beast2_options = create_beast2_options()
)

check_empty_beastier_folders()
```

---

check\_beast2\_options\_filenames\_differ  
*Check if the filenames in beast2\_options differ*

---

**Description**

Calls stop if not.

**Usage**

```
check_beast2_options_filenames_differ(beast2_options)
```

**Arguments**

beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

nothing

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [check\\_beast2\\_options](#) to check the entire beast2\_options object

---

check\_beast2\_options\_names  
*Check if the beast2\_options, which is a list, has all the elements needed.*

---

**Description**

Calls stop if not.

**Usage**

```
check_beast2_options_names(beast2_options)
```

**Arguments**

beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

nothing

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [check\\_beast2\\_options](#) to check the entire `beast2_options` object

---

check_beast2_path	<i>Checks the BEAST2 .jar path. Will stop if there is a problem with the BEAST2 .jar path.</i>
-------------------	--

---

**Description**

Checks the BEAST2 .jar path. Will stop if there is a problem with the BEAST2 .jar path.

**Usage**

```
check_beast2_path(beast2_path)
```

**Arguments**

beast2_path	name of either a BEAST2 binary file (usually simply <code>beast</code> ) or a BEAST2 jar file (usually has a <code>.jar</code> extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path
-------------	---

**Value**

nothing. Will call `stop` if the BEAST2 .jar path has a problem

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed()) {
  beast2_path <- get_default_beast2_jar_path()
  check_beast2_path(beast2_path)
}

check_empty_beastier_folders()
```



---

check\_can\_create\_dir\_for\_state\_output\_file  
*Internal function*

---

**Description**

Check if the folder for the state output file can be created. Will [stop](#) otherwise

**Usage**

```
check_can_create_dir_for_state_output_file(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

Nothing. Will [stop](#) if the folder for the state output file cannot be created

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

check_can_create_dir_for_state_output_file(
  beast2_options = create_beast2_options()
)

check_empty_beastier_folders()
```

---

check\_can\_create\_file *Internal function*

---

**Description**

Check that a file can be created at a certain path.

**Usage**

```
check_can_create_file(filename, overwrite = TRUE)
```

**Arguments**

filename        file that may or may not be created  
overwrite       if TRUE, if filename already exists, it will be deleted by this function

**Details**

Will [stop](#) if not. Will [stop](#) if the file already exists. Does so by creating an empty file at the path, and then deleting it.

**Value**

Nothing. Will [stop](#) if a file cannot be created at a certain path.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_can_create_file("my_local_file.txt")
```

---

check\_can\_create\_screenlog\_file  
*Internal function*

---

**Description**

Check if the MCMC's screenlog file can be created. Will [stop](#) if not

**Usage**

```
check_can_create_screenlog_file(beast2_options)
```

**Arguments**

beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

Nothing. Will [stop](#) if the MCMC's screenlog file cannot be created.

**Author(s)**

Richèl J.C. Bilderbeek

---

check\_can\_create\_state\_output\_file  
*Internal function*

---

**Description**

Check if the state output file can be created. Will [stop](#) otherwise

**Usage**

```
check_can_create_state_output_file(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

Nothing. Will [stop](#) if the state output file cannot be created.

**Author(s)**

Richèl J.C. Bilderbeek

---

check\_can\_create\_tracelog\_file  
*Internal function to check if the MCMC's tracelog file can be created.*

---

**Description**

Check if the MCMC's tracelog file can be created. Will [stop](#) if not. If the tracelog file already exists, it is assumed that a new file can be created, by overwriting the existing one.

**Usage**

```
check_can_create_tracelog_file(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

Nothing. Will [stop](#) if the MCMC's tracelog file is absent and cannot be created.

**Author(s)**

Richèl J.C. Bilderbeek

---

check\_can\_create\_treelog\_file  
*Internal function*

---

**Description**

Check if the MCMC's treelog file can be created. Will [stop](#) if not

**Usage**

```
check_can_create_treelog_file(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

Nothing. Will [stop](#) if the MCMC's treelog file is absent and cannot be created.

**Author(s)**

Richèl J.C. Bilderbeek

---

check\_empty\_beastier\_folder  
*Check there are no files in the default [beastier](#) folder*

---

**Description**

Check there are no files in the default [beastier](#) folder. The goal is to make sure no temporary files are left undeleted. Will [stop](#) if there are files in the [beastier](#) folder

**Usage**

```
check_empty_beastier_folder(beastier_folder = get_beastier_folder())
```

**Arguments**

`beastier_folder`  
the path to the [beastier](#) temporary files folder

**Value**

Nothing. Will **stop** if there are files in the **beastier** folder

**Author(s)**

Richèl J.C. Bilderbeek

---

check\_empty\_beastier\_folders

*Check there are no files in the default 'beautier' and 'beastier' folders*

---

**Description**

Check there are no files in the default 'beautier' and 'beastier' folders.

**Usage**

```
check_empty_beastier_folders(  
    beautier_folder = beautier::get_beautier_folder(),  
    beastier_folder = get_beastier_folder()  
)
```

**Arguments**

beautier\_folder  
temporary folder used by **beautier**

beastier\_folder  
the path to the **beastier** temporary files folder

**Details**

The goal is to make sure no temporary files are left undeleted. Will **stop** if there are files in the **beautier** or **beastier** folder.

**Value**

Nothing. Will **stop** if there are files in the **beautier** or **beastier** folder.

**Author(s)**

Richèl J.C. Bilderbeek

check\_input\_filename *Checks the input filename. Will stop if there is a problem with the input filename.*

---

**Description**

Checks the input filename. Will stop if there is a problem with the input filename.

**Usage**

```
check_input_filename(input_filename)
```

**Arguments**

input\_filename the name of a BEAST2 input XML file. This file usually has an .xml extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.

**Value**

Nothing. Will **stop** if the input file is invalid

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

check_input_filename(
  get_beastier_path("beast2_example_output.log")
)

check_empty_beastier_folders()
```

---

check\_input\_filename\_validity  
*Checks the input filename. Will stop if there is a problem with the input filename.*

---

**Description**

Checks the input filename. Will stop if there is a problem with the input filename.

**Usage**

```
check_input_filename_validity(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

nothing. Will call [stop](#) if the input file is invalid

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed()) {
  check_input_filename_validity(
    create_beast2_options(
      input_filename = get_beastier_path("2_4.xml")
    )
  )
}
check_empty_beastier_folders()
```

---

check_n_threads	<i>Check if the input is a valid number of threads.</i>
-----------------	---

---

**Description**

Will [stop](#) if not.

**Usage**

```
check_n_threads(n_threads)
```

**Arguments**

`n_threads` the number of computational threads to use. Use [NA](#) to use the BEAST2 default of 1.

**Value**

Nothing. Will [stop](#) if the number of threads is invalid

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

# Can have 1 or more threads
check_n_threads(1)
check_n_threads(2)
# Can have NA threads
check_n_threads(NA)

check_empty_beastier_folders()
```

---

check\_os

*Checks if the operating system is supported*

---

**Description**

Checks if the operating system is supported

**Usage**

```
check_os(os)
```

**Arguments**

os                    name of the operating system, must be unix (Linux, Mac) or win (Windows)

**Value**

Nothing. Will [stop](#) if the OS is unsupported

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

check_os("mac")
check_os("unix")
check_os("win")

check_empty_beastier_folders()
```



---

check_rng_seed	<i>Check if the input is a valid RNG seed.</i>
----------------	--

---

**Description**

Will [stop](#) if not.

**Usage**

```
check_rng_seed(rng_seed)
```

**Arguments**

rng_seed	the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or <a href="#">NA</a> . If rng_seed is <a href="#">NA</a> , BEAST2 will pick a random seed
----------	--

**Value**

Nothing. Will [stop](#) if the RNG seed is invalid

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

# Numbers from 1 and higher are valid RNG seeds
check_rng_seed(1)
check_rng_seed(2)
# Also NA is a valid RNG seed
check_rng_seed(NA)

check_empty_beastier_folders()
```

---

continue_beast2	<i>Continue a BEAST2 run</i>
-----------------	------------------------------

---

**Description**

Continue a BEAST2 run

**Usage**

```
continue_beast2(beast2_options = create_beast2_options())
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

The text sent to STDOUT and STDERR. It will create the file with name `output_state_filenames`

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {
  beast2_options <- create_beast2_options(
    input_filename = get_beastier_path("2_4.xml")
  )
  run_beast2_from_options(beast2_options)
  continue_beast2(beast2_options)
  file.remove(beast2_options$output_state_filename)
  remove_beastier_folders()
}

check_empty_beastier_folders()
```

---

```
create_beast2_continue_cmd_from_options
  Creates the terminal command to run BEAST2 from a
  beast2_options
```

---

**Description**

If the BEAST2 input `.xml` filename or the BEAST2 state `.state.xml` filename contain spaces, these filenames are quoted, so that the command-line interface to BEAST2 correctly parses its arguments

**Usage**

```
create_beast2_continue_cmd_from_options(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

a character vector with the command and arguments to call BEAST2

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed()) {  
  create_beast2_continue_cmd_from_options(  
    beast2_options = create_beast2_options()  
  )  
}
```

---

```
create_beast2_input_file_folder
```

*Create the folder where the BEAST2 input file will be created*

---

**Description**

Create the folder where the BEAST2 input file will be created

**Usage**

```
create_beast2_input_file_folder(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

nothing

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()  
  
beast2_options <- create_beast2_options()  
create_beast2_input_file_folder(beast2_options)  
  
remove_beastier_folders()  
check_empty_beastier_folders()
```

---

create\_beast2\_options *Function to create a set of BEAST2 options.*

---

### Description

These BEAST2 options are the R equivalent of the command-line options.

### Usage

```
create_beast2_options(
  input_filename = create_temp_input_filename(),
  output_state_filename = create_temp_state_filename(),
  rng_seed = NA,
  n_threads = NA,
  use_beagle = FALSE,
  overwrite = TRUE,
  beast2_path = get_default_beast2_path(),
  verbose = FALSE
)
```

### Arguments

input_filename	the name of a BEAST2 input XML file. This file usually has an <code>.xml</code> extension. Use <a href="#">create_temp_input_filename</a> to create a temporary filename with that extension.
output_state_filename	name of the <code>.xml</code> state file to create. Use <a href="#">create_temp_state_filename</a> to create a temporary filename with that extension.
rng_seed	the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or <code>NA</code> . If <code>rng_seed</code> is <code>NA</code> , BEAST2 will pick a random seed
n_threads	the number of computational threads to use. Use <code>NA</code> to use the BEAST2 default of 1.
use_beagle	use BEAGLE if present
overwrite	if <code>TRUE</code> : overwrite the <code>.log</code> and <code>.trees</code> files if one of these exists. If <code>FALSE</code> , BEAST2 will not be started if <ul style="list-style-type: none"> <li>• the <code>.log</code> file exists</li> <li>• the <code>.trees</code> files exist</li> <li>• the <code>.log</code> file created by BEAST2 exists</li> <li>• the <code>.trees</code> files created by BEAST2 exist</li> </ul>
beast2_path	name of either a BEAST2 binary file (usually simply <code>beast</code> ) or a BEAST2 jar file (usually has a <code>.jar</code> extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path
verbose	if <code>TRUE</code> , additional information is displayed, that is potentially useful in debugging

**Value**

a BEAST2 options structure, which is a [list](#) of all function arguments, of which all elements are checked (by [check\\_beast2\\_options](#))

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

beast2_options <- create_beast2_options()
check_beast2_options(beast2_options)

check_empty_beastier_folders()
```

---

create\_beast2\_run\_cmd *Creates the terminal command to run BEAST2*

---

**Description**

Creates the terminal command to run BEAST2

**Usage**

```
create_beast2_run_cmd(  
  input_filename,  
  output_state_filename,  
  rng_seed = NA,  
  n_threads = NA,  
  use_beagle = FALSE,  
  overwrite = FALSE,  
  beast2_path = get_default_beast2_path(),  
  verbose = FALSE  
)
```

**Arguments**

input_filename	the name of a BEAST2 input XML file. This file usually has an .xml extension. Use <a href="#">create_temp_input_filename</a> to create a temporary filename with that extension.
output_state_filename	name of the BEAST2 output file that stores the state (usually has a .xml.state extension)
rng_seed	the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or <a href="#">NA</a> . If rng_seed is <a href="#">NA</a> , BEAST2 will pick a random seed

n_threads	the number of computational threads to use. Use <a href="#">NA</a> to use the BEAST2 default of 1.
use_beagle	use BEAGLE if present
overwrite	if TRUE: overwrite the .log and .trees files if one of these exists. If FALSE, BEAST2 will not be started if <ul style="list-style-type: none"> <li>• the .log file exists</li> <li>• the .trees files exist</li> <li>• the .log file created by BEAST2 exists</li> <li>• the .trees files created by BEAST2 exist</li> </ul>
beast2_path	name of either a BEAST2 binary file (usually simply beast) or a BEAST2 jar file (usually has a .jar extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging

**Value**

a character vector with the command and arguments to call BEAST2

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```

check_empty_beastier_folders()

if (is_beast2_installed()) {
  create_beast2_run_cmd(
    input_filename = "input.xml",
    output_state_filename = "output.xml.state",
    beast2_path = get_default_beast2_jar_path()
  )
}

check_empty_beastier_folders()

```

---

```
create_beast2_run_cmd_from_options
```

*Creates the terminal command to run BEAST2 from a  
beast2\_options*

---

**Description**

Creates the terminal command to run BEAST2 from a `beast2_options`

**Usage**

```
create_beast2_run_cmd_from_options(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

a character vector with the command and arguments to call BEAST2

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed()) {
  create_beast2_run_cmd_from_options(
    beast2_options = create_beast2_options()
  )
}

check_empty_beastier_folders()
```

---

```
create_beast2_screenlog_folder
```

*Internal function*

---

**Description**

Create the folder for the BEAST2 screenlog file

**Usage**

```
create_beast2_screenlog_folder(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Author(s)**

Richèl J.C. Bilderbeek

---

create\_beast2\_state\_output\_file\_folder

*Create the folder where the BEAST2 state output file will be created*

---

### Description

Create the folder where the BEAST2 state output file will be created

### Usage

```
create_beast2_state_output_file_folder(beast2_options)
```

### Arguments

beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

### Value

nothing

### Author(s)

Richèl J.C. Bilderbeek

### Examples

```
check_empty_beastier_folders()

beast2_options <- create_beast2_options()
create_beast2_state_output_file_folder(beast2_options)

remove_beastier_folders()
check_empty_beastier_folders()
```

---

create\_beast2\_tracelog\_folder

*Internal function*

---

### Description

Create the folder for the BEAST2 tracelog file

### Usage

```
create_beast2_tracelog_folder(beast2_options)
```



**Arguments**

beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Author(s)**

Richèl J.C. Bilderbeek

---

create\_beast2\_treelog\_folder  
*Internal function*

---

**Description**

Create the folder for the BEAST2 treelog file

**Usage**

```
create_beast2_treelog_folder(beast2_options)
```

**Arguments**

beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Author(s)**

Richèl J.C. Bilderbeek

---

create\_beast2\_validate\_cmd  
*Creates the terminal command to validate a BEAST2 input file*

---

**Description**

Creates the terminal command to validate a BEAST2 input file

**Usage**

```
create_beast2_validate_cmd(  
  input_filename,  
  beast2_path = get_default_beast2_path()  
)
```

**Arguments**

- input\_filename the name of a BEAST2 input XML file. This file usually has an .xml extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.
- beast2\_path name of either a BEAST2 binary file (usually simply beast) or a BEAST2 jar file (usually has a .jar extension). Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default BEAST binary file's path Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default BEAST jar file's path

**Value**

a character vector, of which the first element is the command (java, in this case), and the others are arguments (-jar, in this case, followed by more arguments).

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {
  create_beast2_validate_cmd(
    input_filename = "input.xml"
  )
}

check_empty_beastier_folders()
```

---

```
create_beast2_validate_cmd_bin
```

*Creates the terminal command to validate a BEAST2 input file using a call to the launcher.jar file*

---

**Description**

Creates the terminal command to validate a BEAST2 input file using a call to the launcher.jar file

**Usage**

```
create_beast2_validate_cmd_bin(
  input_filename,
  beast2_bin_path = get_default_beast2_bin_path()
)
```

**Arguments**

- `input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.
- `beast2_bin_path` name of the BEAST2 binary file (usually simply `beast`). Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default BEAST binary file's path

**Value**

a character vector, of which the first element is the command (`java`, in this case), and the others are arguments (`-jar`, in this case, followed by more arguments).

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {
  create_beast2_validate_cmd_bin(
    input_filename = "input.xml"
  )
}

check_empty_beastier_folders()
```

---

```
create_beast2_validate_cmd_jar
```

*Creates the terminal command to validate a BEAST2 input file using a call to the launcher.jar file*

---

**Description**

Creates the terminal command to validate a BEAST2 input file using a call to the `launcher.jar` file

**Usage**

```
create_beast2_validate_cmd_jar(
  input_filename,
  beast2_jar_path = get_default_beast2_jar_path()
)
```

**Arguments**

- `input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.
- `beast2_jar_path` name of the BEAST2 jar file (usually has a `.jar` extension). Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default BEAST jar file's path

**Value**

a character vector, of which the first element is the command (`java`, in this case), and the others are arguments (`-jar`, in this case, followed by more arguments).

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {
  create_beast2_validate_cmd_jar(
    input_filename = "input.xml"
  )
}

check_empty_beastier_folders()
```

---

```
create_beast2_version_cmd
```

*Creates the terminal command to version a BEAST2 input file*

---

**Description**

Creates the terminal command to version a BEAST2 input file

**Usage**

```
create_beast2_version_cmd(beast2_path = beastier::get_default_beast2_path())
```

**Arguments**

- `beast2_path` name of either a BEAST2 binary file (usually simply `beast`) or a BEAST2 jar file (usually has a `.jar` extension). Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default BEAST binary file's path Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default BEAST jar file's path

**Value**

a character vector, of which the first element is the command (java, in this case), and the others are arguments (-jar, in this case, followed by more arguments).

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed()) {  
  create_beast2_version_cmd()  
}
```

---

```
create_beast2_version_cmd_bin
```

*Creates the terminal command to version a BEAST2 input file using a call to the launcher .jar file*

---

**Description**

Creates the terminal command to version a BEAST2 input file using a call to the launcher .jar file

**Usage**

```
create_beast2_version_cmd_bin(beast2_bin_path = get_default_beast2_bin_path())
```

**Arguments**

beast2\_bin\_path

name of the BEAST2 binary file (usually simply beast). Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default BEAST binary file's path

**Value**

a character vector, of which the first element is the command (java, in this case), and the others are arguments (-jar, in this case, followed by more arguments).

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed() && beautier::is_on_ci()) {  
  create_beast2_version_cmd_bin()  
}
```

---

```
create_beast2_version_cmd_jar
```

*Creates the terminal command to version a BEAST2 input file using a call to the launcher . jar file*

---

### Description

Creates the terminal command to version a BEAST2 input file using a call to the launcher . jar file

### Usage

```
create_beast2_version_cmd_jar(beast2_jar_path = get_default_beast2_jar_path())
```

### Arguments

beast2\_jar\_path

name of the BEAST2 jar file (usually has a . jar extension). Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default BEAST jar file's path

### Value

a character vector, of which the first element is the command (java, in this case), and the others are arguments (-jar, in this case, followed by more arguments).

### Author(s)

Richèl J.C. Bilderbeek

### Examples

```
if (is_beast2_installed()) {
  create_beast2_version_cmd_jar()
}
```

---

```
create_beastier_tempfolder
```

*Create the temporary folder as used by [beastier](#)*

---

### Description

Create the temporary folder as used by [beastier](#)

### Usage

```
create_beastier_tempfolder()
```

**Value**

nothing

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()
```

```
create_beastier_tempfolder()
```

```
remove_beastier_folders()
```

```
check_empty_beastier_folders()
```

---

```
create_mcbette_beast2_options
```

*Create a 'beast2\_options' structure for the 'mcbette' R package*

---

**Description**

Create a 'beast2\_options' structure to be used for the 'mcbette' R package, which is a package that allows one to do model comparison. The generated filenames indicating 'mcbette' usage, as well as the correct BEAST2 binary/executable type

**Usage**

```
create_mcbette_beast2_options(  
  input_filename = beastier::create_temp_input_filename(),  
  output_state_filename = beastier::create_temp_state_filename(),  
  rng_seed = NA,  
  n_threads = NA,  
  use_beagle = FALSE,  
  overwrite = TRUE,  
  beast2_bin_path = beastier::get_default_beast2_bin_path(),  
  verbose = FALSE  
)
```

**Arguments**

`input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.

`output_state_filename` name of the `.xml` state file to create. Use [create\\_temp\\_state\\_filename](#) to create a temporary filename with that extension.

rng_seed	the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or <code>NA</code> . If <code>rng_seed</code> is <code>NA</code> , BEAST2 will pick a random seed
n_threads	the number of computational threads to use. Use <code>NA</code> to use the BEAST2 default of 1.
use_beagle	use BEAGLE if present
overwrite	if TRUE: overwrite the <code>.log</code> and <code>.trees</code> files if one of these exists. If FALSE, BEAST2 will not be started if <ul style="list-style-type: none"> <li>• the <code>.log</code> file exists</li> <li>• the <code>.trees</code> files exist</li> <li>• the <code>.log</code> file created by BEAST2 exists</li> <li>• the <code>.trees</code> files created by BEAST2 exist</li> </ul>
beast2_bin_path	name of the BEAST2 binary file (usually simply <code>beast</code> ). Use <code>get_default_beast2_bin_path</code> to get the default BEAST binary file's path
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging

**Value**

a 'beast2\_options' structure suitable to be used by the 'mcbette' R package, which is a [list](#) of all function arguments, of which all elements are checked (by [check\\_beast2\\_options](#))

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

to create a regular (that is, not intended for model comparison) BEAST2 options structure, use [create\\_beast2\\_options](#)

**Examples**

```
check_empty_beastier_folders()
create_mcbette_beast2_options()
check_empty_beastier_folders()
```



---

```
create_random_alignment
```

*Create a random alignment*

---

## Description

Create a random alignment

## Usage

```
create_random_alignment(n_taxa, sequence_length, rate = 1, taxa_name_ext = "")
```

## Arguments

n_taxa	The number of taxa
sequence_length	The number of base pairs the alignment will have
rate	mutation rate
taxa_name_ext	the extension of the taxa names

## Value

an alignment of class [DNABin](#)

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
check_empty_beaustier_folders()

alignment <- create_random_alignment(
  n_taxa = 5,
  sequence_length = 10
)
image(alignment)

check_empty_beaustier_folders()
```

---

create\_random\_fasta    *Create a random FASTA file*

---

### Description

Create a random FASTA file

### Usage

```
create_random_fasta(  
  n_taxa,  
  sequence_length,  
  fasta_filename,  
  taxa_name_ext = ""  
)
```

### Arguments

n\_taxa            The number of taxa  
sequence\_length    a DNA sequence length, in base pairs  
fasta\_filename    a FASTA filename.  
taxa\_name\_ext     the extension of the taxa names

### Value

Nothing, creates a FASTA file

### Author(s)

Richèl J.C. Bilderbeek

### Examples

```
check_empty_beastier_folders()  
  
fasta_filename <- get_beastier_tempfilename()  
create_random_fasta(  
  n_taxa = 5,  
  sequence_length = 20,  
  fasta_filename = fasta_filename  
)  
file.remove(fasta_filename)  
  
remove_beastier_folders()  
check_empty_beastier_folders()
```

---

create\_random\_phylogeny  
*Create a random phylogeny*

---

**Description**

Create a random phylogeny

**Usage**

```
create_random_phylogeny(n_taxa, taxa_name_ext = "")
```

**Arguments**

n\_taxa            The number of taxa  
taxa\_name\_ext    the extension of the taxa names

**Value**

a phylogeny of class 'phylo' (which is part of the 'ape' package)

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
create_random_phylogeny(n_taxa = 6)
```

---

create\_temp\_input\_filename  
*Create a temporary filename for the BEAST2 XML filename*

---

**Description**

Create a temporary filename for the BEAST2 XML filename

**Usage**

```
create_temp_input_filename()
```

**Value**

a temporary filename, that starts with 'beast2\_' and has extension '.xml'

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()
```

```
create_temp_input_filename()
```

```
check_empty_beastier_folders()
```

---

create\_temp\_state\_filename

*Create a temporary file for the BEAST2 XML output file that stores its state.*

---

**Description**

Create a temporary file for the BEAST2 XML output file that stores its state.

**Usage**

```
create_temp_state_filename()
```

**Value**

a temporary filename, that starts with 'beast2\_' and has extension '.xml.state'

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()
```

```
create_temp_state_filename()
```

```
check_empty_beastier_folders()
```

---

default_params_doc	<i>This function does nothing. It is intended to inherit is parameters' documentation.</i>
--------------------	--

---

### Description

This function does nothing. It is intended to inherit is parameters' documentation.

### Usage

```
default_params_doc(  
    beast2_bin_path,  
    beast2_folder,  
    beast2_jar_path,  
    beast2_options,  
    beast2_optionses,  
    beast2_path,  
    beast2_version,  
    beast2_working_dir,  
    beastier_folder,  
    beautier_folder,  
    clock_model,  
    clock_models,  
    crown_age,  
    crown_ages,  
    fasta_filename,  
    fasta_filenames,  
    fixed_crown_age,  
    fixed_crown_ages,  
    initial_phylogenies,  
    input_filename,  
    mcmc,  
    misc_options,  
    n_taxa,  
    n_threads,  
    os,  
    output_filename,  
    output_log_filename,  
    output_state_filename,  
    output_trees_filenames,  
    overwrite,  
    rename_fun,  
    rng_seed,  
    sequence_length,  
    site_model,  
    site_models,  
    tree_prior,
```

```

    tree_priors,
    use_beagle,
    verbose
)

```

## Arguments

beast2_bin_path	name of the BEAST2 binary file (usually simply <code>beast</code> ). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path
beast2_folder	the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a subfolder. Use <a href="#">get_default_beast2_folder</a> to get the default BEAST2 folder. Use <a href="#">get_default_beast2_bin_path</a> to get the full path to the default BEAST2 executable.
beast2_jar_path	name of the BEAST2 jar file (usually has a <code>.jar</code> extension). Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path
beast2_options	a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by <a href="#">create_beast2_options</a>
beast2_optionses	list of one or more <code>beast2_options</code> structures, as can be created by <a href="#">create_beast2_options</a> . Use of reduplicated plural to achieve difference with <code>beast2_options</code>
beast2_path	name of either a BEAST2 binary file (usually simply <code>beast</code> ) or a BEAST2 jar file (usually has a <code>.jar</code> extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path
beast2_version	the version of BEAST2. By default, this is the version as returned by <a href="#">get_default_beast2_version</a>
beast2_working_dir	a folder where BEAST2 can work in isolation. For each BEAST2 run, a new subfolder is created in that folder. Within this folder, BEAST2 is allowed to create all of its output files, without the risk of overwriting existing ones, allowing BEAST2 to run in multiple parallel processes.
beastier_folder	the path to the <a href="#">beastier</a> temporary files folder
beautier_folder	temporary folder used by <a href="#">beautier</a>
clock_model	a <a href="#">beautier</a> clock model
clock_models	a list of one or more <a href="#">beautier</a> clock models
crown_age	the crown age of the phylogeny
crown_ages	the crown ages of the phylogenies. Set to NA if the crown age needs to be estimated
fasta_filename	a FASTA filename.
fasta_filenames	One or more FASTA filenames.

fixed_crown_age	determines if the phylogeny's crown age is fixed. If FALSE, crown age is estimated by BEAST2. If TRUE, the crown age is fixed to the crown age of the initial phylogeny.
fixed_crown_ages	one or more booleans to determine if the phylogenies' crown ages are fixed. If FALSE, crown age is estimated by BEAST2. If TRUE, the crown age is fixed to the crown age of the initial phylogeny.
initial_phylogenies	one or more MCMC chain's initial phylogenies. Each one set to NA will result in BEAST2 using a random phylogeny. Else the phylogeny is assumed to be of class <code>ape::phylo</code> .
input_filename	the name of a BEAST2 input XML file. This file usually has an <code>.xml</code> extension. Use <a href="#">create_temp_input_filename</a> to create a temporary filename with that extension.
mcmc	one <code>beautier</code> MCMC
misc_options	one <code>beautier</code> <code>misc_options</code> object
n_taxa	The number of taxa
n_threads	the number of computational threads to use. Use <code>NA</code> to use the BEAST2 default of 1.
os	name of the operating system, must be <code>unix</code> (Linux, Mac) or <code>win</code> (Windows)
output_filename	Name of the XML parameter file created by this function. BEAST2 uses this file as input.
output_log_filename	name of the <code>.log</code> file to create
output_state_filename	name of the <code>.xml</code> state file to create. Use <a href="#">create_temp_state_filename</a> to create a temporary filename with that extension.
output_trees_filenames	one or more names for <code>.trees</code> file to create. There will be one <code>.trees</code> file created per alignment in the input file. The number of alignments must equal the number of <code>.trees</code> filenames, else an error is thrown. Alignments are sorted alphabetically by their IDs
overwrite	if TRUE: overwrite the <code>.log</code> and <code>.trees</code> files if one of these exists. If FALSE, BEAST2 will not be started if <ul style="list-style-type: none"> <li>• the <code>.log</code> file exists</li> <li>• the <code>.trees</code> files exist</li> <li>• the <code>.log</code> file created by BEAST2 exists</li> <li>• the <code>.trees</code> files created by BEAST2 exist</li> </ul>
rename_fun	a function to rename a filename, as can be checked by <a href="#">check_rename_fun</a> . This function should have one argument, which will be a filename or <code>NA</code> . The function should <code>return</code> one filename (when passed one filename) or one <code>NA</code> (when passed one <code>NA</code> ). Example rename functions are:

- [get\\_remove\\_dir\\_fun](#) get a function that removes the directory paths from the filenames, in effect turning these into local files
- [get\\_replace\\_dir\\_fun](#) get a function that replaces the directory paths from the filenames
- [get\\_remove\\_hex\\_fun](#) get a function that removes the hex string from filenames. For example, `tracelog_82c1a522040.log` becomes `tracelog.log`

rng\_seed           the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or `NA`. If `rng_seed` is `NA`, BEAST2 will pick a random seed

sequence\_length   a DNA sequence length, in base pairs

site\_model        a beautier site model

site\_models       one or more beautier site models

tree\_prior        a beautier tree prior

tree\_priors       one or more beautier tree priors

use\_beagle        use BEAGLE if present

verbose           if TRUE, additional information is displayed, that is potentially useful in debugging

**Value**

Nothing. This is an internal function that does nothing

**Note**

This is an internal function, so it should be marked with `@noRd`. This is not done, as this will disallow all functions to find the documentation parameters

**Author(s)**

Richèl J.C. Bilderbeek

---

do\_minimal\_run

*Do a minimal BEAST2 run*

---

**Description**

To achieve this, [run\\_beast2\\_from\\_options](#) is called.

**Usage**

`do_minimal_run()`

**Value**

The text sent to `STDOUT` and `STDERR`. It will create the files with name `output_state_filename`



**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed() && beautier::is_on_ci()) {  
  do_minimal_run()  
}
```

---

`extract_screenlog_filename_from_beast2_input_file`

*Internal function to extract the screenlog filename for a BEAST2 input file*

---

**Description**

Extract the screenlog filename from a BEAST2 input file

**Usage**

```
extract_screenlog_filename_from_beast2_input_file(input_filename)
```

**Arguments**

`input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.

**Value**

the screenlog filename for a BEAST2 input file

**Author(s)**

Richèl J.C. Bilderbeek

---

extract\_tracelog\_filename\_from\_beast2\_input\_file

*Internal function to extract the tracelog filename for a BEAST2 input file*

---

## Description

Extract the tracelog filename for a BEAST2 input file

## Usage

```
extract_tracelog_filename_from_beast2_input_file(input_filename)
```

## Arguments

`input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.

## Value

the name of the tracelog file

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
if (beautier::is_on_ci())
{
  beast2_input_filename <- get_beastier_tempfilename()
  tracelog_filename <- get_beastier_tempfilename()
  beautier::create_beast2_input_file_from_model(
    input_filename = beautier::get_beautier_path("test_output_0.fas"),
    output_filename = beast2_input_filename,
    inference_model = beautier::create_inference_model(
      mcmc = beautier::create_mcmc(
        tracelog = beautier::create_tracelog(
          filename = tracelog_filename
        )
      )
    )
  )
  extract_tracelog_filename_from_beast2_input_file(
    input_filename = beast2_input_filename
  )
  file.remove(beast2_input_filename)
```

```
    remove_beastier_folders()
  }
```

---

```
extract_treelog_filename_from_beast2_input_file
```

*Internal function to extract the treelog filename for a BEAST2 input file*

---

## Description

Extract the treelog filename from a BEAST2 input file

## Usage

```
extract_treelog_filename_from_beast2_input_file(input_filename)
```

## Arguments

`input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.

## Value

the treelog filename for a BEAST2 input file

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
if (beautier::is_on_ci()) {

  beast2_input_filename <- get_beastier_tempfilename()

  beautier::create_beast2_input_file_from_model(
    input_filename = beautier::get_beautier_path("test_output_0.fas"),
    output_filename = beast2_input_filename
  )
  extract_treelog_filename_from_beast2_input_file(
    input_filename = beast2_input_filename
  )
  file.remove(beast2_input_filename)

  remove_beastier_folders()
}
```

get\_alignment\_ids\_from\_xml\_filename

*Get the alignment ID from a file with one alignment*

---

### **Description**

Get the alignment ID from a file with one alignment

### **Usage**

```
get_alignment_ids_from_xml_filename(xml_filename)
```

### **Arguments**

xml\_filename    name of a BEAST2 XML input filename

### **Value**

one or more alignment IDs

### **Author(s)**

Richèl J.C. Bilderbeek

### **Examples**

```
check_empty_beastier_folders()

# test_output_0
get_alignment_ids_from_xml_filename(get_beastier_path("2_4.xml"))
# c("anthus_aco", "anthus_nd2")
get_alignment_ids_from_xml_filename(get_beastier_path("anthus_15_15.xml"))

check_empty_beastier_folders()
```

---

get\_beast2\_example\_filename

*Get the full path of a BEAST2 example file*

---

### **Description**

Will [stop](#) if the filename is not a BEAST2 example file

**Usage**

```
get_beast2_example_filename(  
    filename,  
    beast2_folder = get_default_beast2_folder()  
)
```

**Arguments**

**filename** name of the BEAST2 example file. This should exclude the full path; this function exists to add that full path

**beast2\_folder** the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a subfolder. Use [get\\_default\\_beast2\\_folder](#) to get the default BEAST2 folder. Use [get\\_default\\_beast2\\_bin\\_path](#) to get the full path to the default BEAST2 executable.

**Value**

the full path of a BEAST2 example file, will [stop](#) if the filename is not a BEAST2 example file

**Examples**

```
if (is_beast2_installed()) {  
  get_beast2_example_filename("testJukesCantor.xml")  
}
```

---

get\_beast2\_example\_filenames

*Get a list with the full paths of all BEAST2 example filenames*

---

**Description**

Get a list with the full paths of all BEAST2 example filenames

**Usage**

```
get_beast2_example_filenames(beast2_folder = get_default_beast2_folder())
```

**Arguments**

**beast2\_folder** the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a subfolder. Use [get\\_default\\_beast2\\_folder](#) to get the default BEAST2 folder. Use [get\\_default\\_beast2\\_bin\\_path](#) to get the full path to the default BEAST2 executable.

**Value**

a list with the full paths of all BEAST2 example filenames

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed()) {  
  get_beast2_example_filenames()  
}
```

---

`get_beast2_main_class_name`

*Get the BEAST2 main class name.*

---

**Description**

One way to fix the error no main manifest attribute is to specify the main class name.

**Usage**

```
get_beast2_main_class_name()
```

**Value**

the BEAST2 main class name

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
get_beast2_main_class_name()
```

---

```
get_beast2_options_filenames
```

*Extract the filenames from a 'beast2\_options'*

---

**Description**

Extract the filenames from a 'beast2\_options'

**Usage**

```
get_beast2_options_filenames(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

the filenames from a 'beast2\_options'

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
beast2_options <- create_beast2_options()
get_beast2_options_filenames(beast2_options)
```

---

```
get_beast2_version
```

*Get the BEAST2 version*

---

**Description**

Get the BEAST2 version

**Usage**

```
get_beast2_version(beast2_path = get_default_beast2_path())
```

**Arguments**

`beast2_path` name of either a BEAST2 binary file (usually simply `beast`) or a BEAST2 jar file (usually has a `.jar` extension). Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default BEAST binary file's path Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default BEAST jar file's path

**Value**

the BEAST2 version

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {
  get_beast2_version()
}

check_empty_beastier_folders()
```

---

get\_beastier\_folder    *Get the path to the [beastier](#) temporary files folder*

---

**Description**

Get the path to the [beastier](#) temporary files folder.

**Usage**

```
get_beastier_folder()
```

**Value**

the path to the [beastier](#) temporary files folder.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
get_beastier_folder()
```



---

get\_beastier\_path      *Get the full path of a file in the inst/extdata folder*

---

**Description**

Get the full path of a file in the inst/extdata folder

**Usage**

```
get_beastier_path(filename)
```

**Arguments**

filename      the file's name, without the path

**Value**

the full path to the filename. Will stop if the file is absent in the inst/extdata folder

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

for more files, use [get\\_beastier\\_paths](#)

**Examples**

```
get_beastier_path("beast2_example_output.log")
get_beastier_path("beast2_example_output.trees")
get_beastier_path("beast2_example_output.xml")
get_beastier_path("beast2_example_output.xml.state")
```

---

get\_beastier\_paths      *Get the full paths of files in the inst/extdata folder*

---

**Description**

Get the full paths of files in the inst/extdata folder

**Usage**

```
get_beastier_paths(filenamees)
```

**Arguments**

filenames        the files' names, without the path

**Value**

the filenames' full paths. Will stop if a file is absent in the inst/extdata folder

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

for one file, use [get\\_beastier\\_path](#)

**Examples**

```
get_beastier_paths(  
  c(  
    "beast2_example_output.log",  
    "beast2_example_output.trees",  
    "beast2_example_output.xml",  
    "beast2_example_output.xml.state"  
  )  
)
```

---

get\_beastier\_tempfilename

*Get a temporary filename*

---

**Description**

Get a temporary filename, similar to [tempfile](#), except that it always writes to a temporary folder named [beastier](#).

**Usage**

```
get_beastier_tempfilename(pattern = "file", fileext = "")
```

**Arguments**

pattern            a non-empty character vector giving the initial part of the name.  
fileext            a non-empty character vector giving the file extension

**Value**

name for a temporary file

**Note**

this function is added to make sure no temporary cache files are left undeleted

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
get_beastier_tempfilename()  
get_beastier_tempfilename(pattern = "my_pattern_")  
get_beastier_tempfilename(fileext = ".ext")
```

---

```
get_default_beast2_bin_path
```

*Get the default BEAST2 binary file (beast, that is) path*

---

**Description**

Get the default BEAST2 binary file (beast, that is) path

**Usage**

```
get_default_beast2_bin_path(  
  beast2_folder = get_default_beast2_folder(),  
  os = rappdirs::app_dir()$os  
)
```

**Arguments**

**beast2\_folder** the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a sub-folder. Use [get\\_default\\_beast2\\_folder](#) to get the default BEAST2 folder. Use [get\\_default\\_beast2\\_bin\\_path](#) to get the full path to the default BEAST2 executable.

**os** name of the operating system, must be unix (Linux, Mac) or win (Windows)

**Value**

the default BEAST2 binary file's path

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [get\\_default\\_beast2\\_folder](#) to get the default folder in which BEAST2 is installed. Use [install\\_beast2](#) with default arguments to install BEAST2 to this location.

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed()) {
  get_default_beast2_bin_path()
}

check_empty_beastier_folders()
```

---

`get_default_beast2_download_url`

*Get the default BEAST2 download URL, which depends on the operating system*

---

**Description**

Get the default BEAST2 download URL, which depends on the operating system

**Usage**

```
get_default_beast2_download_url(
  beast2_version = beastier::get_default_beast2_version(),
  os = rappdirs::app_dir()$os
)
```

**Arguments**

`beast2_version` the version of BEAST2. By default, this is the version as returned by [get\\_default\\_beast2\\_version](#)  
`os` name of the operating system, must be `unix` (Linux, Mac) or `win` (Windows)

**Value**

the URL where BEAST2 can be downloaded from

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
get_default_beast2_download_url()
```

---

`get_default_beast2_download_url_linux`  
*Get the BEAST2 download URL for Linux*

---

### **Description**

Get the BEAST2 download URL for Linux

### **Usage**

```
get_default_beast2_download_url_linux(  
    beast2_version = beastier::get_default_beast2_version()  
)
```

### **Arguments**

`beast2_version` the version of BEAST2. By default, this is the version as returned by [get\\_default\\_beast2\\_version](#)

### **Value**

the URL where BEAST2 can be downloaded from

### **Author(s)**

Richèl J.C. Bilderbeek

---

`get_default_beast2_download_url_win`  
*Get the BEAST2 download URL for Windows*

---

### **Description**

Get the BEAST2 download URL for Windows

### **Usage**

```
get_default_beast2_download_url_win(  
    beast2_version = beastier::get_default_beast2_version()  
)
```

### **Arguments**

`beast2_version` the version of BEAST2. By default, this is the version as returned by [get\\_default\\_beast2\\_version](#)

### **Value**

the URL where BEAST2 can be downloaded from

**Author(s)**

Richèl J.C. Bilderbeek

---

`get_default_beast2_folder`

*Get the path to the folder where this package installs BEAST2 by default*

---

**Description**

Get the path to the folder where this package installs BEAST2 by default

**Usage**

```
get_default_beast2_folder()
```

**Value**

the path to the folder where this package installs BEAST2 by default

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [get\\_default\\_beast2\\_jar\\_path](#) to get the path to the BEAST2 jar file, when installed by this package Use [install\\_beast2](#) with default arguments to install BEAST2 to this folder.

**Examples**

```
check_empty_beastier_folders()
```

```
get_default_beast2_folder()
```

```
check_empty_beastier_folders()
```

---

```
get_default_beast2_jar_path
```

*Get the default BEAST2 jar file's path*

---

### Description

Get the default BEAST2 jar file's path

### Usage

```
get_default_beast2_jar_path(  
    beast2_folder = beastier::get_default_beast2_folder(),  
    os = rappdirs::app_dir()$os  
)
```

### Arguments

`beast2_folder` the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a sub-folder. Use [get\\_default\\_beast2\\_folder](#) to get the default BEAST2 folder. Use [get\\_default\\_beast2\\_bin\\_path](#) to get the full path to the default BEAST2 executable.

`os` name of the operating system, must be `unix` (Linux, Mac) or `win` (Windows)

### Value

the default BEAST2 jar file's path

### Author(s)

Richèl J.C. Bilderbeek

### See Also

Use [get\\_default\\_beast2\\_folder](#) to get the default folder in which BEAST2 is installed. Use [install\\_beast2](#) with default arguments to install BEAST2 to this location.

### Examples

```
check_empty_beastier_folders()  
  
get_default_beast2_jar_path()  
  
check_empty_beastier_folders()
```

---

`get_default_beast2_path`*Get the default BEAST2 path*

---

**Description**

Get the default BEAST2 path

**Usage**

```
get_default_beast2_path(  
    beast2_folder = beastier::get_default_beast2_folder(),  
    os = rappdirs::app_dir()$os  
)
```

**Arguments**

`beast2_folder` the folder where the BEAST2 is installed. Note that this is not the folder where the BEAST2 executable is installed: the BEAST2 executable is in a sub-folder. Use [get\\_default\\_beast2\\_folder](#) to get the default BEAST2 folder. Use [get\\_default\\_beast2\\_bin\\_path](#) to get the full path to the default BEAST2 executable.

`os` name of the operating system, must be `unix` (Linux, Mac) or `win` (Windows)

**Value**

the default BEAST2 path

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [get\\_default\\_beast2\\_bin\\_path](#) to get the default path to the BEAST2 binary file. Use [get\\_default\\_beast2\\_jar\\_path](#) to get the default path to the BEAST2 jar file. Use [get\\_default\\_beast2\\_folder](#) to get the default folder in which BEAST2 is installed. Use [install\\_beast2](#) with default arguments to install BEAST2 to this location.

**Examples**

```
if (is_beast2_installed()) {  
  get_default_beast2_path()  
}
```



---

get\_default\_beast2\_version

*Get the default BEAST2 version that is used by beastier*

---

### **Description**

Get the default BEAST2 version that is used by beastier

### **Usage**

```
get_default_beast2_version()
```

### **Value**

the BEAST2 version

### **Author(s)**

Richèl J.C. Bilderbeek

### **Examples**

```
check_empty_beastier_folders()
```

```
get_default_beast2_version()
```

```
check_empty_beastier_folders()
```

---

get\_default\_java\_path *Obtains the default path to the Java executable*

---

### **Description**

Obtains the default path to the Java executable

### **Usage**

```
get_default_java_path(os = rappdirs::app_dir())$os)
```

### **Arguments**

os                    name of the operating system, must be unix (Linux, Mac) or win (Windows)

### **Value**

the default path to the Java executable

**Author(s)**

Richèl J.C. Bilderbeek

---

get\_duplicate\_param\_ids

*Find duplicate RealParameter IDs*

---

**Description**

Find duplicate RealParameter IDs

**Usage**

```
get_duplicate_param_ids(text)
```

**Arguments**

text            the XML as text

**Value**

a vector of duplicate IDs, will be empty if all IDs are unique

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

to see if all IDs are unique, use [has\\_unique\\_ids](#)

**Examples**

```
check_empty_beustier_folders()

line_1 <- "<parameter id=\"RealParameter.1\" ...</parameter>"
line_2 <- "<parameter id=\"RealParameter.2\" ...</parameter>"
# No elements
get_duplicate_param_ids(c(line_1, line_2))

# 'RealParameter.1'
get_duplicate_param_ids(c(line_1, line_1))

# 'RealParameter.2'
get_duplicate_param_ids(c(line_2, line_2))

check_empty_beustier_folders()
```

---

get_java_version	<i>Get the Java version</i>
------------------	-----------------------------

---

**Description**

Get the Java version

**Usage**

```
get_java_version()
```

**Value**

the Java version

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_beast2_installed() && beautier::is_on_ci()) {  
  get_java_version()  
}
```

---

get_trees_filenames	<i>Get the .trees filenames that BEAST2 will produce</i>
---------------------	--

---

**Description**

Get the .trees filenames that BEAST2 will produce

**Usage**

```
get_trees_filenames(input_filename)
```

**Arguments**

`input_filename` the name of a BEAST2 input XML file. This file usually has an `.xml` extension. Use [create\\_temp\\_input\\_filename](#) to create a temporary filename with that extension.

**Value**

character vector with the names of the .trees files that BEAST2 will produce

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

get_trees_filenames(get_beastier_path("2_4.xml"))
get_trees_filenames(get_beastier_path("anthus_2_4.xml"))

check_empty_beastier_folders()
```

---

`gives_beast2_warning` *Determines if BEAST2 issues a warning when using the BEAST2 XML input file*

---

**Description**

Determines if BEAST2 issues a warning when using the BEAST2 XML input file

**Usage**

```
gives_beast2_warning(
  filename,
  verbose = FALSE,
  beast2_path = beastier::get_default_beast2_path()
)
```

**Arguments**

<code>filename</code>	name of the BEAST2 XML input file
<code>verbose</code>	if TRUE, additional information is displayed, that is potentially useful in debugging
<code>beast2_path</code>	name of either a BEAST2 binary file (usually simply <code>beast</code> ) or a BEAST2 jar file (usually has a <code>.jar</code> extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path

**Value**

TRUE if the file produces a BEAST2 warning, FALSE if not

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [is\\_beast2\\_input\\_file](#) to check if a file is a valid BEAST2 input file. Use [are\\_beast2\\_input\\_lines](#) to check if the text (for example, as loaded from a file) to be valid BEAST2 input.

**Examples**

```
if (is_beast2_installed() &&
    beautier::is_on_ci() &&
    rappdirs::app_dir()$os == "unix") {

    # This file is OK for BEAST2, no warning, returns FALSE
    gives_beast2_warning(filename = get_beastier_path("2_4.xml"))

    # BEAST2 will give a warning on this file, returns TRUE
    gives_beast2_warning(
      filename = get_beastier_path("beast2_warning.xml")
    )
}
```

---

has_unique_ids	<i>Determine if the XML text has unique parameter IDs</i>
----------------	---

---

**Description**

Determine if the XML text has unique parameter IDs

**Usage**

```
has_unique_ids(text)
```

**Arguments**

text            the XML as text

**Value**

TRUE if all parameter IDs are unique, FALSE otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

to obtain the duplicate parameter IDs, use [get\\_duplicate\\_param\\_ids](#)

**Examples**

```

check_empty_beastier_folders()

line_1 <- "<parameter id=\"RealParameter.1\" ...</parameter>"
line_2 <- "<parameter id=\"RealParameter.2\" ...</parameter>"
# Unique IDs
has_unique_ids(c(line_1, line_2))
# No unique ID
has_unique_ids(c(line_1, line_1))

check_empty_beastier_folders()

```

---

install\_beast2

*Deprecated function to install BEAST2*


---

**Description**

This function is deprecated as it violated CRAN policy.

**Usage**

```

install_beast2(
  folder_name = rappdirs::user_data_dir(),
  beast2_version = beastier::get_default_beast2_version(),
  verbose = FALSE,
  os = rappdirs::app_dir()$os
)

```

**Arguments**

folder_name	name of the folder where the BEAST2 files will be put. The name of the BEAST2 binary file will be at [folder_name]/beast/bin/beast The name of the BEAST2 jar file will be at [folder_name]/beast/lib/launcher.jar
beast2_version	the version of BEAST2. By default, this is the version as returned by <a href="#">get_default_beast2_version</a>
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging
os	name of the operating system, must be unix (Linux, Mac) or win (Windows)

**Value**

Nothing. Gives a deprecation message using [stop](#).

**Author(s)**

Richèl J.C. Bilderbeek

---

is_alignment	<i>Determines if the input is an alignment of type <a href="#">DNABin</a></i>
--------------	---

---

**Description**

Determines if the input is an alignment of type [DNABin](#)

**Usage**

```
is_alignment(input)
```

**Arguments**

input	The input to be tested
-------	------------------------

**Value**

TRUE or FALSE

**Author(s)**

Richèl J.C. Bilderbeek

---

is_beast2_input_file	<i>Is a file a valid BEAST2 input file?</i>
----------------------	---

---

**Description**

Is a file a valid BEAST2 input file?

**Usage**

```
is_beast2_input_file(  
  filename,  
  show_warnings = FALSE,  
  verbose = FALSE,  
  beast2_path = get_default_beast2_path()  
)
```

**Arguments**

filename	name of the BEAST2 XML input file
show_warnings	if TRUE, warnings will shown
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging
beast2_path	name of either a BEAST2 binary file (usually simply <code>beast</code> ) or a BEAST2 jar file (usually has a <code>.jar</code> extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path

**Value**

TRUE if the file is valid, FALSE if not

**Note**

this function only works on standard BEAST2 input files: if a BEAST2 input file is modified to use a certain BEAST2 package, this function will label it as an invalid file

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use [are\\_beast2\\_input\\_lines](#) to check the lines

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {

  filename <- get_beastier_path("anthus_2_4.xml")
  # TRUE, this is a BEAST2 input file
  is_beast2_input_file(filename)

  filename <- get_beastier_path("beast2_example_output.log")
  # FALSE, this is not a BEAST2 input file,
  # it is a BEAST2 output log file insteaf
  is_beast2_input_file(filename)
}

check_empty_beastier_folders()
```



---

is\_beast2\_installed     *Checks if BEAST2 is installed*

---

### Description

Checks if BEAST2 is installed

### Usage

```
is_beast2_installed(
  folder_name = get_default_beast2_folder(),
  os = rappdirs::app_dir()$os
)
```

### Arguments

folder_name	name of the folder where the BEAST2 files are put. The name of the BEAST2 binary file will be at [folder_name]/beast/bin/beast The name of the BEAST2 jar file will be at [folder_name]/beast/lib/launcher.jar
os	name of the operating system, must be unix (Linux, Mac) or win (Windows)

### Value

TRUE if BEAST2 is installed

### Author(s)

Richèl J.C. Bilderbeek

### Examples

```
check_empty_beastier_folders()

is_beast2_installed()

check_empty_beastier_folders()
```

---

is_bin_path	<i>Is the path a path to the BEAST2 binary file? Does not check if the file at that path is present</i>
-------------	---

---

### Description

Is the path a path to the BEAST2 binary file? Does not check if the file at that path is present

**Usage**

```
is_bin_path(path)
```

**Arguments**

path                    a string to a path

**Value**

TRUE if the path is a path to a BEAST2 binary file

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

if (is_beast2_installed()) {
  # TRUE
  is_bin_path("beast")
  is_bin_path("BEAST.exe")
  is_bin_path(get_default_beast2_bin_path())
  # FALSE
  is_bin_path("launcher.jar")
  is_bin_path(get_default_beast2_jar_path())
}

check_empty_beastier_folders()
```

---

is_jar_path	<i>Is the path a path to the BEAST2 jar file? Does not check if the file at that path is present</i>
-------------	--

---

**Description**

Is the path a path to the BEAST2 jar file? Does not check if the file at that path is present

**Usage**

```
is_jar_path(path)
```

**Arguments**

path                    a string to a path

**Value**

TRUE if the path is a path to a BEAST2 jar file

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
# Returns TRUE
is_jar_path("beast.jar")
is_jar_path("launcher.jar")
is_jar_path(get_default_beast2_jar_path())
# Returns FALSE
is_jar_path("beast")
is_jar_path(get_default_beast2_bin_path())
```

---

is\_on\_appveyor

*Deprecated function, use [is\\_on\\_appveyor](#)*

---

**Description**

Deprecated function, use [is\\_on\\_appveyor](#)

**Usage**

```
is_on_appveyor()
```

**Value**

TRUE if run on AppVeyor, FALSE otherwise

**Author(s)**

Richèl J.C. Bilderbeek

---

is_on_ci	<i>Deprecated function, use <a href="#">is_on_ci</a></i>
----------	--

---

**Description**

Deprecated function, use [is\\_on\\_ci](#)

**Usage**

```
is_on_ci()
```

**Value**

TRUE if run on AppVeyor or Travis CI, FALSE otherwise

**Author(s)**

Richèl J.C. Bilderbeek

---

is_on_travis	<i>Deprecated function, use <a href="#">is_on_travis</a></i>
--------------	--

---

**Description**

Deprecated function, use [is\\_on\\_travis](#)

**Usage**

```
is_on_travis()
```

**Value**

TRUE if run on Travis CI, FALSE otherwise

**Author(s)**

Richèl J.C. Bilderbeek

---

is_win_bin_path	<i>Is the path a path to the BEAST2 binary file? Does not check if the file at that path is present</i>
-----------------	---

---

**Description**

Is the path a path to the BEAST2 binary file? Does not check if the file at that path is present

**Usage**

```
is_win_bin_path(path)
```

**Arguments**

path            a string to a path

**Value**

TRUE if the path is a path to a BEAST2 binary file

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

# TRUE
is_win_bin_path("BEAST.exe")
# FALSE
is_win_bin_path("beast")
is_win_bin_path("launcher.jar")

check_empty_beastier_folders()
```

---

print_beast2_options	<i>Pretty-print a 'beast2_options'</i>
----------------------	--

---

**Description**

Pretty-print a 'beast2\_options'

**Usage**

```
print_beast2_options(beast2_options)
```

**Arguments**

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

**Value**

Nothing. Will display the ‘beast2\_options’ using [cat](#).

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()
print_beast2_options(create_beast2_options())
check_empty_beastier_folders()
```

---

remove\_beastier\_folder

*Check there are no files in the default [beautier](#) folder*

---

**Description**

Check there are no files in the default [beautier](#) folder. The goal is to make sure no temporary files are left undeleted. Will [stop](#) if there are files in the [beautier](#) folder.

**Usage**

```
remove_beastier_folder()
```

**Value**

Nothing.

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

use [remove\\_beautier\\_folder](#) to remove the default ‘beautier’ folder

**Examples**

```
check_empty_beastier_folder()
remove_beastier_folder()
check_empty_beastier_folder()
```

---

```
remove_beastier_folders
    Remove the 'beautier' and 'beastier' temporary folders
```

---

**Description**

Remove the 'beautier' and 'beastier' temporary folders

**Usage**

```
remove_beastier_folders()
```

**Value**

Nothing.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()
remove_beastier_folders()
check_empty_beastier_folders()
```

---

```
remove_file_if_present
    Remove a file if it is present, will do nothing if it is not.
```

---

**Description**

Remove a file if it is present, will do nothing if it is not.

**Usage**

```
remove_file_if_present(filename)
```

**Arguments**

filename      name of a file

**Value**

Nothing. Will remove the file if it is presented, will do nothing if it is not.

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
filename <- tempfile()
file.create(filename)
remove_file_if_present(filename)
remove_file_if_present(filename)
```

---

rename\_beast2\_options\_filenames

*Rename the filenames in the BEAST2 options*

---

**Description**

Rename the filenames in the BEAST2 options

**Usage**

```
rename_beast2_options_filenames(beast2_options, rename_fun)
```

**Arguments**

- beast2\_options a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)
- rename\_fun a function to rename a filename, as can be checked by [check\\_rename\\_fun](#). This function should have one argument, which will be a filename or `NA`. The function should **return** one filename (when passed one filename) or one `NA` (when passed one `NA`). Example rename functions are:
- [get\\_remove\\_dir\\_fun](#) get a function that removes the directory paths from the filenames, in effect turning these into local files
  - [get\\_replace\\_dir\\_fun](#) get a function that replaces the directory paths from the filenames
  - [get\\_remove\\_hex\\_fun](#) get a function that removes the hex string from filenames. For example, `tracelog_82c1a522040.log` becomes `tracelog.log`



**Value**

a 'beast2\_options' with the filenames it contains renamed

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
check_empty_beastier_folders()

# beast2_options with local filenames
beast2_options <- create_beast2_options(
  input_filename = "my.fas",
  output_state_filename = "my_state.xml.state"
)
# Rename filenames to be in /my/new/folder
rename_beast2_options_filenames(
  beast2_options = beast2_options,
  rename_fun = beautier::get_replace_dir_fun("/my/new/folder")
)

check_empty_beastier_folders()
```

---

run\_beast2

*Run BEAST2*

---

**Description**

Run BEAST2

**Usage**

```
run_beast2(
  input_filename,
  output_state_filename = create_temp_state_filename(),
  rng_seed = NA,
  n_threads = NA,
  use_beagle = FALSE,
  overwrite = TRUE,
  beast2_path = get_default_beast2_path(),
  verbose = FALSE
)
```

**Arguments**

input_filename	the name of a BEAST2 input XML file. This file usually has an .xml extension. Use <a href="#">create_temp_input_filename</a> to create a temporary filename with that extension.
output_state_filename	name of the .xml.state file to create. Use <a href="#">create_temp_state_filename</a> to create a temporary filename with that extension.
rng_seed	the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or <b>NA</b> . If rng_seed is <b>NA</b> , BEAST2 will pick a random seed
n_threads	the number of computational threads to use. Use <b>NA</b> to use the BEAST2 default of 1.
use_beagle	use BEAGLE if present
overwrite	if TRUE: overwrite the .log and .trees files if one of these exists. If FALSE, BEAST2 will not be started if <ul style="list-style-type: none"> <li>• the .log file exists</li> <li>• the .trees files exist</li> <li>• the .log file created by BEAST2 exists</li> <li>• the .trees files created by BEAST2 exist</li> </ul>
beast2_path	name of either a BEAST2 binary file (usually simply beast) or a BEAST2 jar file (usually has a .jar extension). Use <a href="#">get_default_beast2_bin_path</a> to get the default BEAST binary file's path Use <a href="#">get_default_beast2_jar_path</a> to get the default BEAST jar file's path
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging

**Value**

The text sent to STDOUT and STDERR. It will create the file with name output\_state\_filenames

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```

check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {

  output_state_filename <- create_temp_state_filename()

  run_beast2(
    input_filename = get_beastier_path("2_4.xml"),
    output_state_filename = output_state_filename
  )
}

```

```
    file.remove(output_state_filename)
    remove_beastier_folders()
  }

  check_empty_beastier_folders()
```

---

```
run_beast2_from_options
```

```
    Run BEAST2
```

---

## Description

Run BEAST2

## Usage

```
run_beast2_from_options(beast2_options = create_beast2_options())
```

## Arguments

`beast2_options` a set of BEAST2 options, that are the R equivalent of the BEAST2 command-line options, as can be created by [create\\_beast2\\_options](#)

## Value

The text sent to STDOUT and STDERR. It will create the file with name `output_state_filenames`

## Author(s)

Richèl J.C. Bilderbeek

## Examples

```
check_empty_beastier_folders()

if (is_beast2_installed() && beautier::is_on_ci()) {
  beast2_options <- create_beast2_options(
    input_filename = get_beastier_path("2_4.xml")
  )
  run_beast2_from_options(beast2_options)
  file.remove(beast2_options$output_state_filename)
  remove_beastier_folders()
}
check_empty_beastier_folders()
```

---

save_lines	<i>Save text (a container of strings) to a file</i>
------------	---

---

**Description**

Save text (a container of strings) to a file

**Usage**

```
save_lines(filename, lines)
```

**Arguments**

filename	filename of the file to have the text written to
lines	lines of text to be written to file

**Value**

Nothing. Will save the lines to file

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
text <- c("hello", "world")
filename <- get_beastier_tempfilename()
save_lines(filename = filename, lines = text)
file.remove(filename)

remove_beastier_folders()
```

---

save_nexus_as_fasta	<i>Save a NEXUS file as a FASTA file</i>
---------------------	--

---

**Description**

Save a NEXUS file as a FASTA file

**Usage**

```
save_nexus_as_fasta(nexus_filename, fasta_filename)
```

**Arguments**

nexus\_filename name of an existing NEXUS file  
 fasta\_filename name of the FASTA file to be created

**Value**

nothing. The NEXUS file will be saved as a FASTA file

---

uninstall_beast2	<i>Deprecated function to uninstall BEAST2</i>
------------------	--

---

**Description**

Deprecated function to uninstall BEAST2

**Usage**

```
uninstall_beast2(  
  folder_name = rappdirs::user_data_dir(),  
  os = rappdirs::app_dir()$os,  
  verbose = FALSE  
)
```

**Arguments**

folder\_name name of the folder where the BEAST2 files are installed. The name of the BEAST2 binary file will be at [folder\_name]/beast/bin/beast The name of the BEAST2 jar file will be at [folder\_name]/beast/lib/launcher.jar

os name of the operating system, must be unix (Linux, Mac) or win (Windows)

verbose if TRUE, additional information is displayed, that is potentially useful in debugging

**Value**

Nothing. A deprecation message using [stop](#) will be triggered

**Author(s)**

Richèl J.C. Bilderbeek

---

upgrade_beast2	<i>Deprecated function to upgrade BEAST2.</i>
----------------	---

---

**Description**

Deprecated function to upgrade BEAST2.

**Usage**

```
upgrade_beast2(  
  folder_name = rappdirs::user_data_dir(),  
  os = rappdirs::app_dir()$os  
)
```

**Arguments**

folder_name	name of the folder where the BEAST2 files will be put. The name of the BEAST2 binary file will be at [folder_name]/beast/bin/beast The name of the BEAST2 jar file will be at [folder_name]/beast/lib/launcher.jar
os	name of the operating system, must be unix (Linux, Mac) or win (Windows)

**Value**

Nothing. A deprecation message using [stop](#) will be triggered

**Author(s)**

Richèl J.C. Bilderbeek

# Index

add\_quotes\_if\_has\_spaces, 4  
are\_beast2\_input\_lines, 5, 69, 72  
are\_beast2\_input\_lines\_deep, 6  
are\_beast2\_input\_lines\_fast, 7  
are\_identical\_alignments, 8  
  
beast2\_options\_to\_table, 8  
beastier, 9, 10, 20, 21, 38, 46, 56, 58  
beastier\_report, 10  
beautier, 21, 46, 78  
  
cat, 78  
check\_beast2, 11  
check\_beast2\_options, 11, 13, 15, 16, 29, 40  
check\_beast2\_options\_data\_types, 13  
check\_beast2\_options\_do\_not\_overwrite\_existing\_files, 14  
check\_beast2\_options\_filenames\_differ, 15  
check\_beast2\_options\_names, 15  
check\_beast2\_optionses, 12  
check\_beast2\_path, 16  
check\_can\_create\_dir\_for\_state\_output\_file, 17  
check\_can\_create\_file, 17  
check\_can\_create\_screenlog\_file, 18  
check\_can\_create\_state\_output\_file, 19  
check\_can\_create\_tracelog\_file, 19  
check\_can\_create\_treelog\_file, 20  
check\_empty\_beastier\_folder, 20  
check\_empty\_beastier\_folders, 21  
check\_input\_filename, 22  
check\_input\_filename\_validity, 22  
check\_n\_threads, 23  
check\_os, 24  
check\_rename\_fun, 47, 80  
check\_rng\_seed, 25  
continue\_beast2, 25  
create\_beast2\_continue\_cmd\_from\_options, 26  
create\_beast2\_input\_file\_folder, 27  
create\_beast2\_options, 9, 11–15, 17–20, 23, 26, 27, 28, 31–33, 40, 46, 55, 78, 80, 83  
create\_beast2\_run\_cmd, 29  
create\_beast2\_run\_cmd\_from\_options, 30  
create\_beast2\_screenlog\_folder, 31  
create\_beast2\_state\_output\_file\_folder, 32  
create\_beast2\_tracelog\_folder, 32  
create\_beast2\_treelog\_folder, 33  
create\_beast2\_validate\_cmd, 33  
create\_beast2\_validate\_cmd\_bin, 34  
create\_beast2\_validate\_cmd\_jar, 35  
create\_beast2\_version\_cmd, 36  
create\_beast2\_version\_cmd\_bin, 37  
create\_beast2\_version\_cmd\_jar, 38  
create\_beastier\_tempfolder, 38  
create\_mcbette\_beast2\_options, 39  
create\_random\_alignment, 41  
create\_random\_fasta, 42  
create\_random\_phylogeny, 43  
create\_temp\_input\_filename, 22, 28, 29, 34–36, 39, 43, 47, 49–51, 67, 82  
create\_temp\_state\_filename, 28, 39, 44, 47, 82  
  
default\_params\_doc, 45  
DNABin, 41, 71  
do\_minimal\_run, 48  
  
extract\_screenlog\_filename\_from\_beast2\_input\_file, 49  
extract\_tracelog\_filename\_from\_beast2\_input\_file, 50  
extract\_treelog\_filename\_from\_beast2\_input\_file, 51  
  
get\_alignment\_ids\_from\_xml\_filename, 52

- get\_beast2\_example\_filename, 52
- get\_beast2\_example\_filenames, 53
- get\_beast2\_main\_class\_name, 54
- get\_beast2\_options\_filenames, 55
- get\_beast2\_version, 55
- get\_beastier\_folder, 56
- get\_beastier\_path, 57, 58
- get\_beastier\_paths, 57, 57
- get\_beastier\_tempfilename, 58
- get\_default\_beast2\_bin\_path, 5, 6, 10, 11, 16, 28, 30, 34–37, 40, 46, 53, 55, 59, 59, 63, 64, 68, 72, 82
- get\_default\_beast2\_download\_url, 60
- get\_default\_beast2\_download\_url\_linux, 61
- get\_default\_beast2\_download\_url\_win, 61
- get\_default\_beast2\_folder, 10, 46, 53, 59, 60, 62, 63, 64
- get\_default\_beast2\_jar\_path, 5, 6, 11, 16, 28, 30, 34, 36, 38, 46, 55, 62, 63, 64, 68, 72, 82
- get\_default\_beast2\_path, 64
- get\_default\_beast2\_version, 46, 60, 61, 65, 70
- get\_default\_java\_path, 65
- get\_duplicate\_param\_ids, 66, 69
- get\_java\_version, 67
- get\_remove\_dir\_fun, 48, 80
- get\_remove\_hex\_fun, 48, 80
- get\_replace\_dir\_fun, 48, 80
- get\_trees\_filenames, 67
- gives\_beast2\_warning, 68
  
- has\_unique\_ids, 66, 69
  
- install\_beast2, 60, 62–64, 70
- is\_alignment, 71
- is\_beast2\_input\_file, 6, 7, 69, 71
- is\_beast2\_installed, 73
- is\_bin\_path, 73
- is\_jar\_path, 74
- is\_on\_appveyor, 75, 75
- is\_on\_ci, 76, 76
- is\_on\_travis, 76, 76
- is\_win\_bin\_path, 77
  
- list, 29, 40
  
- message, 10
  
- NA, 23, 25, 28–30, 40, 47, 48, 80, 82
  
- print\_beast2\_options, 77
  
- remove\_beastier\_folder, 78
- remove\_beastier\_folders, 79
- remove\_beastier\_folder, 78
- remove\_file\_if\_present, 79
- rename\_beast2\_options\_filenames, 80
- return, 47, 80
- run\_beast2, 81
- run\_beast2\_from\_options, 48, 83
  
- save\_lines, 84
- save\_nexus\_as\_fasta, 84
- stop, 11, 14, 16–25, 52, 53, 70, 78, 85, 86
  
- tempfile, 58
- tibble, 9
  
- uninstall\_beast2, 85
- upgrade\_beast2, 86