Package ‘beastier’

December 2, 2019

Type Package
Title Call ‘BEAST2’
Version 2.1.1
Maintainer Richèl J.C. Bilderbeek <richel@richelbilderbeek.nl>
Description ‘BEAST2’ (<http://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
LazyData true
RoxygenNote 6.1.1
Depends beastier (>= 2.3)
Imports ape, assertive, phangorn, rappdirs, remotes, rJava, stringr, xml2
Suggests hunspell, knitr, 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 (http://www.beast2.org/)
NeedsCompilation no
R topics documented:

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),
Thijs Janzen [ctb]

Repository  CRAN

Date/Publication  2019-12-02 12:50:11 UTC

R topics documented:

are_beast2_input_lines ........................................... 3
are_beast2_input_lines_deep ...................................... 5
are_beast2_input_lines_fast ...................................... 6
are_identical_alignments ......................................... 7
beast2_internal_filenames_to_table  ......................... 7
beast2_options_to_table ........................................ 8
beastier .......................................................... 8
check_beast2 ....................................................... 9
check_beast2_internal_filenames  ............................... 10
check_beast2_options ............................................ 11
check_beast2_optionses .......................................... 12
check_beast2_path ............................................... 13
check_can_create_file ........................................... 13
check_input_filename ............................................ 14
check_n_threads .................................................. 15
check_os .......................................................... 15
check_rng_seed .................................................... 16
create_beast2_internal_filenames  ......................... 17
create_beast2_options ........................................... 18
create_beast2_run_cmd .......................................... 19
create_beast2_validate_cmd .................................. 21
create_beast2_validate_cmd_bin ................................ 22
create_beast2_validate_cmd_jar ................................ 23
create_beast2_version_cmd .................................... 24
create_beast2_version_cmd_bin ................................ 24
create_beast2_version_cmd_jar ................................ 25
create_temp_input_filename .................................... 26
create_temp_state_filename .................................... 26
default_params_doc ............................................. 27
do_minimal_run .................................................... 30
get_alignment_ids_from_xml_filename ......................... 30
get_beast2_example_filename ................................... 31
get_beast2_example_filenames .................................. 32
get_beast2_main_class_name .................................... 33
get_beast2_options_filenames .................................. 33
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?
are_beast2_input_lines

Usage

are_beast2_input_lines(lines, verbose = FALSE,
method = ifelse(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 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

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

library(testthat)

if (is_beast2_installed() && is_on_ci()) {
    beast2_filename <- get_beastier_path("anthus_2.4.xml")
    text <- readLines(beast2_filename)
    expect_true(are_beast2_input_lines(text))
}
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 `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

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() && is_on_ci()) {
  beast2_filename <- get_beastier_path("anthus_2_4.xml")
  text <- readLines(beast2_filename)
  testit::assert(are_beast2_input_lines_deep(text))
}
```
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

library(testthat)

beast2_filename <- get_beastier_path("anthus_2_4.xml")
text <- readLines(beast2_filename)
expect_true(are_beast2_input_lines_fast(text))
are_identical_alignments

Determines if the two alignments are equal

**Description**

Determines if the two alignments are equal

**Usage**

```r
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_internal_filenames_to_table

Convert a `beast2_internal_filenames` to a table

**Description**

Convert a `beast2_internal_filenames` to a table

**Usage**

```r
beast2_internal_filenames_to_table(beast2_internal_filenames)
```

**Arguments**

- `beast2_internal_filenames`: a list of internally used BEAST2 filenames, as created by `create_beast2_internal_filenames`


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

beastier

beastier: A package to call BEAST2.

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

library(testthat)

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

if (is_beast2_installed() && is_on_ci()) {
  expect_false(file.exists(beast2_options$output_state_filename))
}
check_beast2 <- run_beast2_from_options(beast2_options)
expect_true(length(output) > 0)
expect_true(file.exists(beast2_options$output_state_filename))

check_beast2

Check if BEAST2 is installed properly.

Description
Calls stop if BEAST2 is improperly installed

Usage
check_beast2(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
nothing

Author(s)
Richèl J.C. Bilderbeek

Examples

library(testthat)
if (is_beast2_installed()) {
  expect_silent(check_beast2())
} else {
  expect_error(check_beast2())
}
check_beast2_internal_filenames

Check if the beast2_internal_filenames is a valid BEAST2 internal filenames object.

Description

Calls stop if the BEAST2 internal filenames object is invalid

Usage

check_beast2_internal_filenames(beast2_internal_filenames)

Arguments

beast2_internal_filenames

a list of internally used BEAST2 filenames, as created by create_beast2_internal_filenames

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_beast2_internal_filenames to create a valid BEAST2 internal filenames object

Examples

library(testthat)

if (beastier::is_beast2_installed()) {
  expect_silent(
    check_beast2_internal_filenames(
      create_beast2_internal_filenames(
        create_beast2_options(
          input_filename = get_beastier_path("2_4.xml")
        )
      )
    )
  )
}

# Must stop on nonsense
expect_error(check_beast2_internal_filenames("nonsense"))
expect_error(check_beast2_internal_filenames(NULL))
expect_error(check_beast2_internal_filenames(NA))
**check_beast2_options**  
*Check if the beast2_options is a valid BEAST2 options object.*

**Description**

Calls `stop` if the BEAST2 option object is invalid.

**Usage**

```r
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

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use `create_beast2_options` to create a valid BEAST2 options object.

**Examples**

```r
library(testthat)

expect_silent(check_beast2_options(create_beast2_options()))

# Must stop on nonsense
expect_error(check_beast2_options(beast2_options = "nonsense"))
expect_error(check_beast2_options(beast2_options = NULL))
expect_error(check_beast2_options(beast2_options = NA))
```
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

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_beast2_options to create a valid BEAST2 options object

Examples

library(testthat)

expect_silent(check_beast2_optionses(list(create_beast2_options())))

# Must stop on nonsense
expect_error(check_beast2_optionses("nonsense"))
expect_error(check_beast2_optionses(NULL))
expect_error(check_beast2_optionses(NA))
check_beast2_path

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

Description

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

Usage

check_beast2_path(beast2_path)

description

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 call stop if the BEAST2 .jar path has a problem

Author(s)

Richèl J.C. Bilderbeek

Examples

library(testthat)

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

check_can_create_file

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

Description

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.

Usage

check_can_create_file(filename, overwrite = TRUE)
check_input_filename

Arguments

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

Author(s)

Richèl J.C. Bilderbeek

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 call stop if the input file is invalid

Author(s)

Richèl J.C. Bilderbeek

Examples

library(testthat)

expect_error(check_input_filename(tempfile()))
expect_silent(
    check_input_filename(
        get_beastier_path("beast2_example_output.log")
    )
)
check_n_threads

Check if the input is a valid number of threads.

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.

Author(s)
Richèl J.C. Bilderbeek

Examples
library(testthat)
expect_silent(check_n_threads(1))
expect_silent(check_n_threads(2))
expect_silent(check_n_threads(NA))

expect_error(check_n_threads(-1))
expect_error(check_n_threads("nonsense"))
expect_error(check_n_threads(c(1, 2)))
expect_error(check_n_threads(c())))
expect_error(check_n_threads(NULL))

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)
check_rng_seed

Value
nothing. Will stop if the OS is unsupported

Author(s)
Richèl J.C. Bilderbeek

Examples

```r
library(testthat)

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

expect_error(check_os("android"))
expect_error(check_os("n64"))
expect_error(check_os("nds"))
expect_error(check_os("nes"))
expect_error(check_os("snes"))

expect_error(check_os(NA))
expect_error(check_os(NULL))
expect_error(check_os(NA))
expect_error(check_os(c()))
```

Description
Will stop if not.

Usage

```r
check_rng_seed(rng_seed)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rng_seed</code></td>
<td>the random number generator seed of the BEAST2 run. Must be a non-zero positive integer value or NA. If <code>rng_seed</code> is NA, BEAST2 will pick a random seed</td>
</tr>
</tbody>
</table>

Author(s)
Richèl J.C. Bilderbeek
create_beast2_internal_filenames

Create a list with the internally used BEAST2 filenames

Description
Create a list with the internally used BEAST2 filenames

Usage
create_beast2_internal_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
a list with the internally used BEAST2 filenames

Examples
library(testthat)

beast2_options <- create_beast2_options(
  input_filename = get_beastier_path("2_4.xml")
)
if (is_beast2_installed()) {
  beast2_internal_filenames <- create_beast2_internal_filenames(beast2_options)
  bif_names <- names(beast2_internal_filenames)
  expect_true("input_filename_full" %in% bif_names)
  expect_true("output_state_filename_full" %in% bif_names)
}
**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,
output_log_filename = "deprecated",
output_trees_filenames = "deprecated",
beast2_working_dir = "deprecated")
```

**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 NA. If `rng_seed` is NA, BEAST2 will pick a random seed.
- **n_threads** the number of computational threads to use. Use NA 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:
  - the `.log` file exists
  - the `.trees` files exist
  - the `.log` file created by BEAST2 exists
  - the `.trees` files created by BEAST2 exist
- **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.
- **verbose** if TRUE, additional information is displayed, that is potentially useful in debugging.
create_beast2_run_cmd

output_log_filename
  name of the .log file to create
output_trees_filenames
  one or more names for .trees file to create. There will be one .trees file created per alignment in the input file. The number of alignments must equal the number of .trees filenames, else an error is thrown. Alignments are sorted alphabetically by their IDs
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.

Value
  a BEAST2 options structure

Author(s)
  Richèl J.C. Bilderbeek

Examples

library(testthat)

beast2_options <- create_beast2_options()

expect_true("input_filename" %in% names(beast2_options))
expect_true("output_state_filename" %in% names(beast2_options))
expect_true("rng_seed" %in% names(beast2_options))
expect_true("n_threads" %in% names(beast2_options))
expect_true("use_beagle" %in% names(beast2_options))
expect_true("overwrite" %in% names(beast2_options))
expect_true("beast2_path" %in% names(beast2_options))
expect_true("verbose" %in% names(beast2_options))

expect_silent(check_beast2_options(beast2_options))

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())
create_beast2_run_cmd

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 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 NA. If `rng_seed` is NA, BEAST2 will pick a random seed

- **n_threads**
  the number of computational threads to use. Use NA 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
  - the .log file exists
  - the .trees files exist
  - the .log file created by BEAST2 exists
  - the .trees files created by BEAST2 exist

- **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 with the command and arguments to call BEAST2

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
if (is_beast2_installed()) {
  cmds <- create_beast2_run_cmd(
    input_filename = "input.xml",
    output_state_filename = "output.xml.state",
    beast2_path = get_default_beast2_jar_path()
  )
  testit::assert(cmds[2] == "-cp")
}
```
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**

```r
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**

```r
if (is_beast2_installed() && is_on_ci()) {
  cmds <- create_beast2_validate_cmd(
    input_filename = "input.xml"
  )
  testit::assert(cmds[2] == "-cp")
}
```
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

```r
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

```r
if (is_beast2_installed() && is_on_ci()) {
    cmds <- create_beast2_validate_cmd_bin(
        input_filename = "input.xml"
    )
    testit::assert(length(cmds) == 3)
    testit::assert(cmds[2] == "-validate")
}
```
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

   if (is_beast2_installed() && is_on_ci()) {
      cmds <- create_beast2_validate_cmd_jar(
         input_filename = "input.xml"
      )
      testit::assert(length(cmds) == 6)
      testit::assert(cmds[2] == "-cp")
   }
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

```r
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

```r
if (is_beast2_installed() && is_on_ci()) {
  cmds <- create_beast2_version_cmd()
  testit::assert(cmds[2] == "-cp")
}
```

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
create_beast2_version_cmd_bin

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() && is_on_ci()) {
  cmds <- create_beast2_version_cmd_bin()
  testit::assert(length(cmds) == 2)
  testit::assert(cmds[2] == "-version")
}

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
create_temp_state_filename

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

```r
if (is_beast2_installed()) {
  cmds <- create_beast2_version_cmd_jar()
  testit::assert(length(cmds) == 5)
  testit::assert(cmds[2] == "-cp")
}
```

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()
```

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()
```
This function does nothing. It is intended to inherit is parameters’ documentation.

Description

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

Usage

default_params_doc(beast2_bin_path, beast2_folder, beast2_internal_filenames, beast2_jar_path, beast2_options, beast2_optionses, beast2_path, beast2_working_dir, 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, 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 beast). Use get_default_beast2_bin_path 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 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.

beast2_internal_filenames
a list of internally used BEAST2 filenames, as created by create_beast2_internal_filenames

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

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

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

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
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 cre-
ate all of its output files, without the risk of overwriting existing ones, allowing
BEAST2 to run in multiple parallel processes.

clock_model  a beautier clock model
clock_models  a list of one or more beautier 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 es-
timated 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 ape::phylo.
input_filename  the name of a BEAST2 input XML file. This file usually has an .xml exten-
sion. Use create_temp_input_filename to create a temporary filename with that
extension.
mcmc  one beautier MCMC
misc_options  one beautier misc_options object
n_taxa  The number of taxa
n_threads  the number of computational threads to use. Use NA to use the BEAST2 default
of 1.
os  name of the operating system, must be unix (Linux, Mac) or win (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 .log file to create
output_state_filename name of the .xml.state file to create. Use create_temp_state_filename to cre-
ate a temporary filename with that extension.
output_trees_filenames
one or more names for .trees file to create. There will be one .trees file created per alignment in the input file. The number of alignments must equal the number of .trees filenames, else an error is thrown. Alignments are sorted alphabetically by their IDs

overwrite
if TRUE: overwrite the .log and .trees files if one of these exists. If FALSE, BEAST2 will not be started if
• the .log file exists
• the .trees files exist
• the .log file created by BEAST2 exists
• the .trees files created by BEAST2 exist

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_beaagle
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**

```r
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**

```r
library(testthat)

if (is_beast2_installed() && is_on_ci()) {
  expect_silent(do_minimal_run())
}
```

---

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**

```r
get_alignment_ids_from_xml_filename(xml_filename)
```

**Arguments**

- `xml_filename` name of a BEAST2 XML input filename
get_beast2_example_filename

Value

one or more alignment IDs

Author(s)

Richèl J.C. Bilderbeek

Examples

library(testthat)

expect_equal(
  get_alignment_ids_from_xml_filename(get_beastier_path("2_4.xml"),
  "test_output_0"
)
)

expect_equal(
  get_alignment_ids_from_xml_filename(
    get_beastier_path("anthus_15_15.xml")
  ),
  c("anthus_aco","anthus_nd2")
)

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 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.
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 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.

Value
a list with the full paths of all 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()

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

Author(s)

Richèl J.C. Bilderbeek

Examples

library(testthat)

beast2_options <- beastier::create_beast2_options()
filenames <- get_beast2_options_filenames(beast2_options)
expect_true(beast2_options$input_filename %in% filenames)
expect_true(beast2_options$output_state_filename %in% filenames)
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

Author(s)
Richèl J.C. Bilderbeek

Examples
library(testthat)

if (is_beast2_installed() && is_on_ci()) {
  expect_equal(get_beast2_version(), "2.6.0")
}

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
**get_beastier_paths**

**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 one file, use *get_beastier_path*

**Examples**

```r
library(testthat)

expect_true(is.character(get_beastier_path("beast2_example_output.log")))
expect_true(is.character(get_beastier_path("beast2_example_output.trees")))
expect_true(is.character(get_beastier_path("beast2_example_output.xml")))
expect_true(
    is.character(get_beastier_path("beast2_example_output.xml.state"))
)
```

---

**Description**

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

**Usage**

```r
get_beastier_paths(filenames)
```

**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 more files, use *get_beastier_paths*
Examples

library(testthat)

filenames <- get_beastier_paths(
  c(
    "beast2_example_output.log",
    "beast2_example_output.trees",
    "beast2_example_output.xml",
    "beast2_example_output.xml.state"
  )
)

expect_equal(length(filenames), 4)
expect_true(all(file.exists(filenames)))

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**

```r
if (is_beast2_installed() && rappdirs::app_dir()$os == "unix") {
  testit::assert(
    grepl(
      "beast/bin/beast",
      get_default_beast2_bin_path()
    )
  )
}
```

---

**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**

```r
get_default_beast2_download_url(os = rappdirs::app_dir()$os)
```

**Arguments**

- `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**

```r
library(testthat)

eexpect_true(
  grepl(
    "https://github.com/CompEvol/beast2/releases/download",
    get_default_beast2_download_url()
  )
)
```

if (rappdirs::app_dir()$os == "unix") {
  expect_true(
    grepl(
```
get_default_beast2_download_url_win

"BEAST.v[0-9\.|]+.Linux.tgz",
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()

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()

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**

```r
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**

```r
print(get_default_beast2_folder())
```

---

get_default_beast2_jar_path

*Get the default BEAST2 jar file's path*

---

**Description**

Get the default BEAST2 jar file’s path

**Usage**

```r
get_default_beast2_jar_path(os = rappdirs::app_dir()$os)
```

**Arguments**

- **os**
  
  name of the operating system, must be unix (Linux, Mac) or win (Windows)
get_default_beast2_path

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

```r
library(testthat)

if (is_beast2_installed() && rappdirs::app_dir()$os == "unix") {
  expect_true(
    grepl(
      "beast/lib/launcher.jar",
      get_default_beast2_jar_path()
    )
  )
}
```

get_default_beast2_path

*Get the default BEAST2 path*

Description

Get the default BEAST2 path

Usage

```r
get_default_beast2_path()
```

Value

the default BEAST2 path

Author(s)

Richèl J.C. Bilderbeek
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

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

library(testthat)

if (is_beast2_installed()) {
  def_path <- get_default_beast2_path()
  bin_path <- get_default_beast2_bin_path()
  jar_path <- get_default_beast2_jar_path()
  expect_true(def_path == bin_path || def_path == jar_path)
}

-------------------

get_default_java_path  Obtains the default path to the Java executable
-------------------
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**

```r
line_1 <- "<parameter id="RealParameter.1" ...</parameter>"
line_2 <- "<parameter id="RealParameter.2" ...</parameter>"

testit::assert(
  length(get_duplicate_param_ids(c(line_1, line_2))) == 0)

testit::assert(
  get_duplicate_param_ids(
    c(line_1, line_1)) == c("RealParameter.1")
)

testit::assert(
  get_duplicate_param_ids(
    c(line_2, line_2)) == c("RealParameter.2")
)
```
get_java_version  Get the Java version

Description
Get the Java version

Usage
get_java_version()

Author(s)
Richèl J.C. Bilderbeek

Examples
library(testthat)

if (is_beast2_installed() && is_on_ci()) {
  expect_silent(get_java_version())
}

get_trees_filenames  Get the .trees filenames that BEAST2 will produce

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

```r
library(testthat)

expect_equal(
  get_trees_filenames(get_beastier_path("2_4.xml")),
  "test_output_0.trees"
)

expect_equal(
  get_trees_filenames(get_beastier_path("anthus_2_4.xml")),
  c("Anthus_nd2.trees", "Anthus_aco.trees")
)
```

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

```r
gives_beast2_warning(filename, verbose = FALSE,
  beast2_path = get_default_beast2_path())
```

Arguments

- **filename**: name of the BEAST2 XML input file
- **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 `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

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

library(testthat)

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

  # This file is OK for BEAST2
  expect_false(
    gives_beast2_warning(
      filename = get_beastier_path("2_4.xml")
    )
  )
  
  # BEAST2 will give a warning on this file
  expect_true(
    gives_beast2_warning(
      filename = get_beastier_path("beast2_warning.xml")
    )
  )
}

has_unique_ids  Determine if the XML text has unique parameter IDs

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
install_beast2

install_beast2

Install BEAST2

Usage

install_beast2(folder_name = rappdirs::user_data_dir(),
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
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. Will install BEAST2

Author(s)

Richèl J.C. Bilderbeek

Examples

library(testthat)

if (is_on_ci()) {
  if (!is_beast2_installed()) {
    install_beast2()
  }
  expect_true(file.exists(get_default_beast2_path()))
  expect_true(file.exists(get_default_beast2_bin_path()))
  expect_true(file.exists(get_default_beast2_jar_path()))
}

library(testthat)

line_1 <- "<parameter id="RealParameter.1" ...</parameter>
line_2 <- "<parameter id="RealParameter.2" ...</parameter>
expect_true(has_unique_ids(c(line_1, line_2)))
expect_false(has_unique_ids(c(line_1, line_1)))
is_alignment

Determine if the input is an alignment of type **DNAbin**

**Description**

Determines if the input is an alignment of type **DNAbin**

**Usage**

```r
is_alignment(input)
```

**Arguments**

- **input**: The input to be tested

**Value**

TRUE or FALSE

**Author(s)**

Richèl J.C. Bilderbeek

---

is_beast2_input_file

Is a file a valid BEAST2 input file?

**Description**

Is a file a valid BEAST2 input file?

**Usage**

```r
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 `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
is_beast2_installed

Checks if BEAST2 is installed

Usage

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

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folder_name</td>
<td>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.</td>
</tr>
<tr>
<td>os</td>
<td>name of the operating system, must be unix (Linux, Mac) or win (Windows).</td>
</tr>
</tbody>
</table>
is_bin_path

Value

TRUE if BEAST2 is installed

Author(s)

Richèl J.C. Bilderbeek

Examples

if (is_beast2_installed()) {
  print("BEAST2 is installed")
}

---

is_bin_path path

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

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

library(testthat)

if (is_beast2_installed()) {
  expect_true(is_bin_path("beast"))
  expect_true(is_bin_path("BEAST.exe"))
  expect_false(is_bin_path("launcher.jar"))
  expect_true(is_bin_path(get_default_beast2_bin_path()))
  expect_false(is_bin_path(get_default_beast2_jar_path()))
}
**is_jar_path**

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

**Description**

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

**Usage**

```r
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**

```r
library(testthat)

expect_false(is_jar_path("beast"))
expect_true(is_jar_path("beast.jar"))
expect_true(is_jar_path("launcher.jar"))
expect_false(is_jar_path(get_default_beast2_bin_path()))
expect_true(is_jar_path(get_default_beast2_jar_path()))
```

---

**is_on_appveyor**

*Determines if the environment is AppVeyor*

**Description**

Determines if the environment is AppVeyor

**Usage**

```r
is_on_appveyor()
```
is_on_ci

Value

TRUE if run on AppVeyor, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

Examples

if (is_on_appveyor()) {
    print("Running on AppVeyor")
}

is_on_ci

Determines if the environment is a continuous integration service

Description

Determines if the environment is a continuous integration service

Usage

is_on_ci()

Value

TRUE if run on AppVeyor or Travis CI, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

Examples

if (is_on_ci()) {
    print("Running on a continuous integration service")
}
**is_on_travis**

*Determines if the environment is Travis CI*

**Description**

Determines if the environment is Travis CI

**Usage**

```
is_on_travis()
```

**Value**

TRUE if run on Travis CI, FALSE otherwise

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```
if (is_on_travis()) {
  print("Running on Travis CI")
}
```

**print_beast2_internal_filenames**

*Print a beast2_internal_filenames as a table*

**Description**

Print a beast2_internal_filenames as a table

**Usage**

```
print_beast2_internal_filenames(beast2_internal_filenames)
```

**Arguments**

```
beast2_internal_filenames
```

a list of internally used BEAST2 filenames, as created by `create_beast2_internal_filenames`
print_beast2_options  Pretty-print a beast2_options

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

remove_file_if_present  Remove a file if it is present

Description

Remove a file if it is present

Usage

remove_file_if_present(filename)

Arguments

filename  name of a file
run_beast2 | Run BEAST2

Description

Run BEAST2

Usage

```r
run_beast2(input_filename,
            output_log_filename = "output_log_filename_is_deprecated",
            output_trees_filenames = "output_trees_filenames_is_deprecated",
            output_state_filename = create_temp_state_filename(),
            rng_seed = NA,
            n_threads = NA, use_beagle = FALSE, overwrite = TRUE,
            beast2_working_dir = "beast2_working_dir_is_deprecated",
            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 `create_temp_input_filename` to create a temporary filename with that extension.
- `output_log_filename` name of the .log file to create
- `output_trees_filenames` one or more names for .trees file to create. There will be one .trees file created per alignment in the input file. The number of alignments must equal the number of .trees filenames, else an error is thrown. Alignments are sorted alphabetically by their IDs
- `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 NA. If `rng_seed` is NA, BEAST2 will pick a random seed
- `n_threads` the number of computational threads to use. Use NA 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
  - the .log file exists
  - the .trees files exist
  - the .log file created by BEAST2 exists
  - the .trees files created by BEAST2 exist
run_beast2_from_options

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.

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

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
   library(testthat)
   if (is_beast2_installed() && is_on_ci()) {
     output_state_filename <- tempfile(fileext = ".xml.state")
     expect_false(file.exists(output_state_filename))
     output <- run_beast2(
       input_filename = get_beastier_path("2.4.xml"),
       output_state_filename = output_state_filename
     )
     expect_true(length(output) > 40)
     expect_true(file.exists(output_state_filename))
   }

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

Author(s)

Richèl J.C. Bilderbeek

Examples

library(testthat)

if (is_beast2_installed() && is_on_ci()) {
  beast2_options <- create_beast2_options(
    input_filename = get_beastier_path("2_4.xml")
  )

  expect_false(file.exists(beast2_options$output_state_filename))

  output <- run_beast2_from_options(beast2_options)

  expect_true(length(output) > 40)
  expect_true(file.exists(beast2_options$output_state_filename))
}

---

save_lines  \(\text{Save text (a container of strings) to a file}\)

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
save_nexus_as_fasta

Author(s)
Richèl J.C. Bilderbeek

Examples

```r
library(testthat)

text <- c("hello", "world")
filename <- tempfile(fileext = ".txt")

eexpect_false(file.exists(filename))
save_lines(filename = filename, lines = text)
eexpect_true(file.exists(filename))
```

Description

Save a NEXUS file as a FASTA file

Usage

```r
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

uninstall_beast2

Uninstall BEAST2

Description

Uninstall BEAST2

Usage

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

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
library(testthat)

if (is_beast2_installed() && is_on_ci()) {
  uninstall_beast2()
  expect_false(is_beast2_installed())
  install_beast2()
  expect_true(is_beast2_installed())
}
```

update_beastier  
*Update all beastier dependencies, by installing their latest versions*

Description

Update all beastier dependencies, by installing their latest versions

Usage

```r
update_beastier()
```

Author(s)

Richèl J.C. Bilderbeek
**upgrade_beast2**

*Upgrade BEAST2.*

**Description**

Will stop if BEAST2 is not installed

**Usage**

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

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
library(testthat)

if (is_beast2_installed() && is_on_ci()) {
  expect_equal(get_beast2_version(), "2.6.0")
}
```
Index

are_beast2_input_lines, 3, 44, 48
are_beast2_input_lines_deep, 5
are_beast2_input_lines_fast, 6
are_identical_alignments, 7
beast2_internal_filenames_to_table, 7
beast2_options_to_table, 8
beastier, 8
beastier-package (beastier), 8
check_beast2, 9
check_beast2_internal_filenames, 10
check_beast2_options, 11
check_beast2_optiones, 12
check_beast2_path, 13
check_can_create_file, 13
check_input_filename, 14
check_n_threads, 15
check_os, 15
check_rng_seed, 16
create_beast2_internal_filenames, 7, 10, 17, 27, 52
create_beast2_options, 8, 11, 12, 17, 18, 27, 33, 53, 56
create_beast2_run_cmd, 19
create_beast2_validate_cmd, 21
create_beast2_validate_cmd_bin, 22
create_beast2_validate_cmd_jar, 23
create_beast2_version_cmd, 24
create_beast2_version_cmd_bin, 24
create_beast2_version_cmd_jar, 25
create_temp_input_filename, 14, 18, 20–23, 26, 28, 43, 54
create_temp_state_filename, 18, 26, 28, 54
default_params_doc, 27
DNAbin, 47
do_minimal_run, 30
get_alignment_ids_from_xml_filename,
get_beast2_example_filename, 30
get_beast2_example_filenames, 32
get_beast2_main_class_name, 33
get_beast2_options_filenames, 33
get_beast2_version, 34
get_beastier_path, 34, 35
get_beastier_paths, 35, 35
get_default_beast2_bin_path, 4, 5, 9, 13,
18, 20–22, 24, 25, 27, 31, 32, 34, 36, 37, 41, 44, 47, 55
get_default_beast2_download_url, 37
get_default_beast2_download_url_linux, 38
get_default_beast2_download_url_win, 38
get_default_beast2_folder, 27, 31, 32, 36,
39, 40, 41
get_default_beast2_jar_path, 4, 5, 9, 13,
18, 20, 21, 23–25, 27, 34, 39, 39, 41, 44, 47, 55
get_default_beast2_path, 40
get_default_java_path, 41
get_duplicate_param_ids, 42, 45
get_java_version, 43
get_trees_filenames, 43
gives_beast2_warning, 44
has_unique_ids, 42, 45
install_beast2, 36, 39–41, 46
is_alignment, 47
is_beast2_input_file, 4–6, 44, 47
is_beast2_installed, 48
is_bin_path, 49
is_jar_path, 50
is_on_appveyor, 50
is_on_ci, 51
is_on_travis, 52
INDEX

NA, 15, 16, 18, 20, 28, 29, 54

print_beast2_internal_filenames, 52
print_beast2_options, 53

remove_file_if_present, 53
run_beast2, 54
run_beast2_from_options, 30, 55

save_lines, 56
save_nexus_as_fasta, 57
stop, 13–16, 31, 59

uninstall_beast2, 57
update_beastier, 58
upgrade_beast2, 59