Package 'beautier'

March 1, 2019

Title 'BEAUti' from R

Version 2.2.1

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.
'BEAUti 2' (which is part of 'BEAST2') is a GUI tool that allows users to specify the many possible setups and generates the XML file 'BEAST2' needs to run.
This package provides a way to create 'BEAST2' input files without active user input, but using R function calls instead.

License GPL-3 | file LICENSE

LazyData true

RoxygenNote 6.1.1

VignetteBuilder knitr

URL https://github.com/ropensci/beautier

BugReports https://github.com/ropensci/beautier/issues

Imports ape, geiger, seqinr, stringr, testit

Suggests spelling, devtools, knitr, ggplot2, hunspell, lintr, rmarkdown, testthat

Language en-US

Encoding UTF-8

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),

Paul Van Els [ctb],
Raphael Scherrer [ctb],
Yacine B. Chehida [ctb],
Jana Riederer [ctb]

Repository CRAN

Date/Publication 2019-03-01 14:10:12 UTC

R topics documented:

are_clock_models *Determine if x consists out of clock_models objects*

Description

Determine if x consists out of clock_models objects

Usage

```
are_clock_models(x)
```

Arguments

x	the object to check if it consists out of clock_models objects

Value

TRUE if x, or all elements of x, are clock_model objects

Author(s)

Richèl J.C. Bilderbeek

Examples

```
rln_clock_model <- create_rln_clock_model()
strict_clock_model <- create_strict_clock_model()
both_clock_models <- list(rln_clock_model, strict_clock_model)
testit::assert(are_clock_models(rln_clock_model))
testit::assert(are_clock_models(strict_clock_model))
testit::assert(are_clock_models(both_clock_models))
```

are_equal_mcmcs *Determine if two MCMCs are equal.*

Description

Will stop if the arguments are not MCMCs.

Usage

```
are_equal_mcmcs(mcmc_1, mcmc_2)
```

Arguments

mcmc_1	an MCMC, as created by create_mcmc
mcmc_2	an MCMC, as created by create_mcmc

Value

TRUE if the two MCMCs are equal

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_mcmc to create an MCMC

Examples

```
mcmc_1 <- create_mcmc(chain_length = 1000)
mcmc_2 <- create_mcmc(chain_length = 314)
testthat::expect_true(are_equal_mcmcs(mcmc_1, mcmc_1))
testthat::expect_false(are_equal_mcmcs(mcmc_1, mcmc_2))
```

---

are_equivalent_xml_lines

Determine if XML lines result in equivalent trees

Description

Determine if XML lines result in equivalent trees

Usage

```
are_equivalent_xml_lines(lines_1, lines_2, section = NA,
  verbose = FALSE)
```

Arguments

lines_1	lines of a first XML file
lines_2	lines of a second XML file
section	the name of the XML section
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging

Value

TRUE if the two XML lines result in equivalent trees, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

`are_equivalent_xml_lines_all` *Determine if XML lines result in equivalent trees*

Description

Determine if XML lines result in equivalent trees

Usage

```
are_equivalent_xml_lines_all(lines_1, lines_2, verbose = FALSE)
```

Arguments

`lines_1`	lines of a first XML file
`lines_2`	lines of a second XML file
`verbose`	if TRUE, additional information is displayed, that is potentially useful in debugging

Value

TRUE if the two XML lines result in equivalent trees, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

`are_equivalent_xml_lines_loggers` *Determine if XML operator lines result in equivalent trees*

Description

Determine if XML operator lines result in equivalent trees

Usage

```
are_equivalent_xml_lines_loggers(lines_1, lines_2, verbose = FALSE)
```

Arguments

`lines_1`	lines of a first XML file
`lines_2`	lines of a second XML file
`verbose`	if TRUE, additional information is displayed, that is potentially useful in debugging

Value

TRUE if the two XML lines result in equivalent trees, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

`are_equivalent_xml_lines_operators`
Determine if XML operator lines result in equivalent trees

Description

Determine if XML operator lines result in equivalent trees

Usage

```
are_equivalent_xml_lines_operators(lines_1, lines_2, verbose = FALSE)
```

Arguments

`lines_1`	lines of a first XML file
`lines_2`	lines of a second XML file
`verbose`	if TRUE, additional information is displayed, that is potentially useful in debugging

Value

TRUE if the two XML lines result in equivalent trees, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

are_equivalent_xml_lines_section *Determine if XML lines result in equivalent trees*

Description

Determine if XML lines result in equivalent trees

Usage

```
are_equivalent_xml_lines_section(lines_1, lines_2, section,
  verbose = FALSE)
```

Arguments

lines_1	lines of a first XML file
lines_2	lines of a second XML file
section	the name of the XML section
verbose	if TRUE, additional information is displayed, that is potentially useful in debugging

Value

TRUE if the two XML lines result in equivalent trees, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

are_site_models *Determine if x consists out of site_models objects*

Description

Determine if x consists out of site_models objects

Usage

```
are_site_models(x)
```

Arguments

x	the object to check if it consists out of site_models objects

Value

TRUE if x, or all elements of x, are site_model objects

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_model to create a site model

Examples

```
jc69_site_model <- create_jc69_site_model()
gtr_site_model <- create_gtr_site_model()
both_site_models <- list(jc69_site_model, gtr_site_model)
testit::assert(are_site_models(jc69_site_model))
testit::assert(are_site_models(gtr_site_model))
testit::assert(are_site_models(both_site_models))
```

---

are_tree_priors *Determine if x consists out of tree_priors objects*

Description

Determine if x consists out of tree_priors objects

Usage

```
are_tree_priors(x)
```

Arguments

x	the object to check if it consists out of tree_priors objects

Value

TRUE if x, or all elements of x, are tree_prior objects

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_yule_tree_prior to create a Yule tree prior

Examples

```
yule_tree_prior <- create_yule_tree_prior()
bd_tree_prior <- create_bd_tree_prior()
both_tree_priors <- list(yule_tree_prior, bd_tree_prior)
testit::assert(are_tree_priors(yule_tree_prior))
testit::assert(are_tree_priors(bd_tree_prior))
testit::assert(are_tree_priors(both_tree_priors))
```

`bd_tree_prior_to_xml_prior_distr`

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Birth-Death tree prior

---

Description

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Birth-Death tree prior

Usage

```
bd_tree_prior_to_xml_prior_distr(bd_tree_prior)
```

Arguments

`bd_tree_prior`	a Birth-Death tree prior, as created by `create_bd_tree_prior`

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#       HERE, where the ID of the distribution is 'prior'
#     </distribution>
#     <distribution id="likelihood" ...>
#     </distribution>
# </distribution>
```

beautier *beautier: A package to create a BEAST2 input file.*

Description

beautier allows to create a BEAST2 input file, using an R interface. beautier closely follows the interface of BEAUti 2, a GUI tool bundled with BEAST2, including its default settings.

Author(s)

Richèl J.C. Bilderbeek

See Also

These are packages associated with beautier:

- The package beastier can run BEAST2 from R
- The package tracerer can parse BEAST2 output files from R
- The package mauricer manages BEAST2 packages from R
- The package babette combines the functionality of beautier, beastier, mauricer and tracerer into a single workflow

Examples

```
# Get an example FASTA file
input_filename <- get_fasta_filename()

# The file created by beautier, a BEAST2 input file
output_filename <- tempfile()

# Use the default BEAUti settings to create a BEAST2 input file
create_beast2_input_file_from_model(
  input_filename,
  output_filename,
  inference_model = create_inference_model()
)
testthat::expect_true(file.exists(output_filename))
```

cbs_tree_prior_to_xml_prior_distr *Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Birth-Death tree prior*

Description

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Birth-Death tree prior

Usage

```
cbs_tree_prior_to_xml_prior_distr(cbs_tree_prior)
```

Arguments

cbs_tree_prior	a Coalescent Bayesian Skyline tree prior, as returned by create_cbs_tree_prior

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#       HERE, where the ID of the distribution is 'prior'
#     </distribution>
#     <distribution id="likelihood" ...>
#     </distribution>
# </distribution>
```

---

ccp_tree_prior_to_xml_prior_distr

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Coalescent Constant Population tree prior

---

Description

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Coalescent Constant Population tree prior

Usage

```
ccp_tree_prior_to_xml_prior_distr(ccp_tree_prior)
```

Arguments

ccp_tree_prior	a Coalescent Constant Population tree prior, as returned by create_ccp_tree_prior

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#       HERE, where the ID of the distribution is 'prior'
#     </distribution>
#     <distribution id="likelihood" ...>
#     </distribution>
# </distribution>
```

`cep_tree_prior_to_xml_prior_distr`

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Coalescent Exponential Population tree prior

---

Description

Creates the tree prior section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Coalescent Exponential Population tree prior

Usage

```
cep_tree_prior_to_xml_prior_distr(cep_tree_prior)
```

Arguments

`cep_tree_prior`	a Coalescent Exponential Population tree prior, as returned by `create_cep_tree_prior`

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#       HERE, where the ID of the distribution is 'prior'
#     </distribution>
#     <distribution id="likelihood" ...>
#     </distribution>
# </distribution>
```

check_beauti_options *Check if the* beauti_options *is a valid* beauti_options *object.*

Description

Calls stop if the beauti_options object is invalid

Usage

```
check_beauti_options(beauti_options)
```

Arguments

beauti_options	one BEAUti options object, as returned by create_beauti_options

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_beauti_options to create a valid BEAUti options setup

Examples

```
testthat::expect_silent(check_beauti_options(create_beauti_options()))

# Must stop on nonsense
testthat::expect_error(check_beauti_options(beauti_options = "nonsense"))
testthat::expect_error(check_beauti_options(beauti_options = NULL))
testthat::expect_error(check_beauti_options(beauti_options = NA))
```

check_clock_model *Check if the clock model is a valid clock model.*

Description

Calls stop if the clock model is invalid

Usage

```
check_clock_model(clock_model)
```

Arguments

`clock_model` a clock model, as returned by `create_clock_model`

Value

TRUE if `clock_model` is a valid clock model

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_model to create a valid clock model

Examples

```
testthat::expect_silent(check_clock_model(create_strict_clock_model()))
testthat::expect_silent(check_clock_model(create_rln_clock_model()))

# Must stop on non-clock models
testthat::expect_error(check_clock_model(clock_model = "nonsense"))
testthat::expect_error(check_clock_model(clock_model = NULL))
testthat::expect_error(check_clock_model(clock_model = NA))
```

`check_clock_models` *Check if the object is a list of one or more clock models.*

Description

Will stop if the object is not a list of one or more clock models.

Usage

```
check_clock_models(clock_models)
```

Arguments

`clock_models` the object to be checked if it is a list of one or more valid clock models

Value

nothing. Will stop if the object is not a list of one or more clock models.

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_model to create a valid clock model

Examples

```
testthat::expect_silent(check_clock_models(create_strict_clock_model()))
testthat::expect_silent(
  check_clock_models(list(create_strict_clock_model()))
)
testthat::expect_silent(
  check_clock_models(
    list(create_strict_clock_model(), create_rln_clock_model())
  )
)

testthat::expect_error(check_clock_models("nonsense"))
testthat::expect_error(check_clock_models(3.14))
testthat::expect_error(check_clock_models(42))
testthat::expect_error(check_clock_models(NA))
testthat::expect_error(check_clock_models(NULL))
```

---

check_inference_model *Check if the supplied object is a valid Bayesian phylogenetic inference model.*

Description

Calls `stop` if the supplied object is not a valid Bayesian phylogenetic inference model.

Usage

```
check_inference_model(inference_model)
```

Arguments

`inference_model`	an Bayesian phylogenetic inference model, as can be created by `create_inference_model`. An inference model is the complete model setup in which a site model, clock model, tree prior and more are specified.

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_inference_model to create a valid Bayesian phylogenetic inference model

Examples

```
testthat::expect_silent(check_inference_model(create_inference_model()))

# Must stop on non-MCMCs
testthat::expect_error(check_inference_model(inference_model = "nonsense"))
testthat::expect_error(check_inference_model(inference_model = NULL))
testthat::expect_error(check_inference_model(inference_model = NA))
```

---

check_inference_models

Check if the `inference_model` *is a valid BEAUti inference model.*

Description

Calls `stop` if not.

Usage

```
check_inference_models(inference_models)
```

Arguments

`inference_models`	a list of one or more inference models, as can be created by create_inference_model

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_inference_model to create a valid BEAST2 options object

Examples

```
testthat::expect_silent(
  check_inference_models(
      list(create_inference_model()
    )
  )
)

# Must stop on nonsense
testthat::expect_error(check_inference_models("nonsense"))
testthat::expect_error(check_inference_models(NULL))
testthat::expect_error(check_inference_models(NA))
```

---

check_mcmc *Check if the MCMC is a valid MCMC object.*

Description

Calls `stop` if the MCMC is invalid

Usage

```
check_mcmc(mcmc)
```

Arguments

`mcmc`	one MCMC as returned by `create_mcmc`

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_mcmc to create a valid MCMC

Examples

```
testthat::expect_silent(check_mcmc(create_mcmc()))

# Must stop on non-MCMCs
testthat::expect_error(check_mcmc(mcmc = "nonsense"))
testthat::expect_error(check_mcmc(mcmc = NULL))
testthat::expect_error(check_mcmc(mcmc = NA))
```

check_mrca_prior *Check if the MRCA prior is a valid MRCA prior.*

Description

Calls stop if the MRCA prior is invalid.

Usage

```
check_mrca_prior(mrca_prior)
```

Arguments

mrca_prior	a Most Recent Common Ancestor prior, as returned by create_mrca_prior

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_mrca_prior to create a valid MRCA prior

Examples

```
fasta_filename <- get_beautier_path("anthus_aco.fas")
mrca_prior <- create_mrca_prior(
  alignment_id = get_alignment_id(fasta_filename = fasta_filename),
  taxa_names = get_taxa_names(filename = fasta_filename)
)
mrca_prior <- create_mrca_prior(
 alignment_id = get_alignment_id(fasta_filename = fasta_filename),
 taxa_names = get_taxa_names(filename = fasta_filename)
)
testthat::expect_silent(check_mrca_prior(mrca_prior))

# NA is a valid MRCA prior
testthat::expect_silent(check_mrca_prior(mrca_prior = NA))

# Must stop on non-MRCA priors
testthat::expect_error(check_mrca_prior(mrca_prior = "nonsense"))
testthat::expect_error(check_mrca_prior(mrca_prior = NULL))
```

check_phylogeny *Check if the phylogeny is a valid phylogeny object.*

Description

Calls stop if the phylogeny is invalid

Usage

```
check_phylogeny(phylogeny)
```

Arguments

phylogeny	a phylogeny of type phylo

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use read.tree to create a phylogeny

Examples

```
phylogeny <- ape::read.tree(text = "(A:1, B:1):1;")
testthat::expect_silent(check_phylogeny(phylogeny))

# Must stop on non-phylogenies
testthat::expect_error(check_phylogeny(phylo = "nonsense"))
testthat::expect_error(check_phylogeny(phylo = NULL))
testthat::expect_error(check_phylogeny(phylo = NA))
```

check_rln_clock_model *Check if the clock model is a valid clock model.*

Description

Calls stop if the clock model is invalid

Usage

```
check_rln_clock_model(clock_model)
```

Arguments

clock_model	a clock model, as returned by create_clock_model

Value

TRUE if clock_model is a valid clock model

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_model to create a valid clock model

Examples

```
testthat::expect_silent(
  check_rln_clock_model(create_rln_clock_model())
)
testthat::expect_error(
  check_rln_clock_model(create_strict_clock_model())
)
```

check_site_model *Check if the site model is a valid site model*

Description

Calls stop if the site models are invalid

Usage

```
check_site_model(site_model)
```

Arguments

site_model	a site model, as returned by create_site_model

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_model to create a valid site model

Examples

```
testthat::expect_silent(check_site_model(create_jc69_site_model()))
testthat::expect_silent(check_site_model(create_hky_site_model()))
testthat::expect_silent(check_site_model(create_tn93_site_model()))
testthat::expect_silent(check_site_model(create_gtr_site_model()))

# Can use list of one site model
testthat::expect_silent(check_site_model(list(create_jc69_site_model())))

# List of two site models is not a/one site model
testthat::expect_error(
  check_site_model(
    list(create_jc69_site_model(), create_jc69_site_model())
  )
)

# Must stop on non-site models
testthat::expect_error(check_site_model(site_model = "nonsense"))
testthat::expect_error(check_site_model(site_model = NULL))
testthat::expect_error(check_site_model(site_model = NA))
```

---

check_site_models *Check if the object is a list of one or more site models.*

Description

Will stop if the object is not a list of one or more site models.

Usage

```
check_site_models(site_models)
```

Arguments

site_models	the object to be checked if it is a list of one or more valid site models

Value

nothing. Will stop if the object is not a list of one or more site models.

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_model to create a valid site model

Examples

```
testthat::expect_silent(check_site_models(create_jc69_site_model()))
testthat::expect_silent(check_site_models(list(create_jc69_site_model())))
testthat::expect_silent(
  check_site_models(
    list(create_jc69_site_model(), create_gtr_site_model())
  )
)

testthat::expect_error(check_site_models("nonsense"))
testthat::expect_error(check_site_models(3.14))
testthat::expect_error(check_site_models(42))
testthat::expect_error(check_site_models(NA))
testthat::expect_error(check_site_models(NULL))
```

---

check_strict_clock_model

Check if the clock model is a valid clock model.

Description

Calls stop if the clock model is invalid

Usage

```
check_strict_clock_model(clock_model)
```

Arguments

clock_model	a clock model, as returned by create_clock_model

Value

TRUE if `clock_model` is a valid clock model

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_model to create a valid clock model

Examples

```
testthat::expect_silent(
  check_strict_clock_model(create_strict_clock_model())
)
testthat::expect_error(
  check_strict_clock_model(create_rln_clock_model())
)
```

`check_tree_prior` *Check if the tree prior is a valid tree prior*

Description

Calls `stop` if the tree priors are invalid

Usage

```
check_tree_prior(tree_prior)
```

Arguments

`tree_prior`	a tree priors, as returned by `create_tree_prior`

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_tree_prior to create a valid tree prior

Examples

```
testthat::expect_silent(check_tree_prior(create_yule_tree_prior()))
testthat::expect_silent(check_tree_prior(create_bd_tree_prior()))
testthat::expect_silent(check_tree_prior(create_cbs_tree_prior()))
testthat::expect_silent(check_tree_prior(create_ccp_tree_prior()))
testthat::expect_silent(check_tree_prior(create_cep_tree_prior()))

# Can use list of one tree prior
testthat::expect_silent(check_tree_prior(list(create_yule_tree_prior())))

# List of two tree priors is not a/one tree prior
testthat::expect_error(
  check_tree_prior(
    list(create_yule_tree_prior(), create_yule_tree_prior())
  )
)

# Must stop on non-tree priors
testthat::expect_error(check_tree_prior(tree_prior = "nonsense"))
testthat::expect_error(check_tree_prior(tree_prior = NULL))
testthat::expect_error(check_tree_prior(tree_prior = NA))
```

---

check_tree_priors *Check if the object is a list of one or more tree priors.*

Description

Will stop if the object is not a list of one or more tree priors.

Usage

```
check_tree_priors(tree_priors)
```

Arguments

tree_priors	the object to be checked if it is a list of one or more valid tree priors

Value

nothing. Will stop if the object is not a list of one or more tree priors.

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_tree_prior to create a valid tree prior

Examples

```
testthat::expect_silent(check_tree_priors(create_yule_tree_prior()))
testthat::expect_silent(check_tree_priors(list(create_yule_tree_prior())))
testthat::expect_silent(
  check_tree_priors(
    list(create_yule_tree_prior(), create_bd_tree_prior())
  )
)

testthat::expect_error(check_tree_priors("nonsense"))
testthat::expect_error(check_tree_priors(3.14))
testthat::expect_error(check_tree_priors(42))
testthat::expect_error(check_tree_priors(NA))
testthat::expect_error(check_tree_priors(NULL))
```

---

create_alpha_param *Create a parameter called alpha*

Description

Create a parameter called alpha

Usage

```
create_alpha_param(id = NA, value = 0)
```

Arguments

id	the parameter's ID
value	value of the parameter

Value

a parameter called alpha

Note

this parameter is used in a beta distribution (as returned by create_beta_distr) and gamma distribution (as returned by create_gamma_distr) and inverse-gamma distribution (as returned by create_inv_gamma_distr). It cannot be estimated (as a hyper parameter) yet.

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_param contains a list of all parameters that can be created

Examples

```
# Create the parameter
alpha_param <- create_alpha_param()

# Use the parameter in a distribution
beta_distr <- create_beta_distr(
  alpha = alpha_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = beta_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_bd_tree_prior *Create a Birth-Death tree prior*

Description

Create a Birth-Death tree prior

Usage

```
create_bd_tree_prior(
  id = NA,
  birth_rate_distr = create_uniform_distr(),
  death_rate_distr = create_uniform_distr()
)
```

Arguments

`id`	the ID of the alignment
`birth_rate_distr`	the birth rate distribution, as created by a `create_distr` function
`death_rate_distr`	the death rate distribution, as created by a `create_distr` function

Value

a Birth-Death tree_prior

Author(s)

Richèl J.C. Bilderbeek

See Also

An alignment ID can be extracted from its FASTA filename using `get_alignment_id`

Examples

```
bd_tree_prior <- create_bd_tree_prior()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = bd_tree_prior
)
testit::assert(file.exists(beast2_input_file))

bd_tree_prior_exp <- create_bd_tree_prior(
  birth_rate_distr = create_exp_distr()
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = bd_tree_prior_exp
)
testit::assert(file.exists(beast2_input_file))
```

---

create_beast2_input *Create a BEAST2 XML input text*

Description

Create a BEAST2 XML input text

Usage

```
create_beast2_input(input_filename, tipdates_filename = NA,
  site_model = create_jc69_site_model(),
  clock_model = create_strict_clock_model(),
  tree_prior = create_yule_tree_prior(), mrca_prior = NA,
  mcmc = create_mcmc(), beauti_options = create_beauti_options(),
  input_filenames = "deprecated", site_models = "deprecated",
  clock_models = "deprecated", tree_priors = "deprecated",
  mrca_priors = "deprecated", posterior_crown_age = "deprecated")
```

Arguments

input_filename	A FASTA filename. Use get_fasta_filename to obtain a testing FASTA filename.
tipdates_filename	name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.
site_model	a site model, as returned by create_site_model
clock_model	a clock model, as returned by create_clock_model
tree_prior	a tree priors, as returned by create_tree_prior
mrca_prior	a Most Recent Common Ancestor prior, as returned by create_mrca_prior
mcmc	one MCMC as returned by create_mcmc
beauti_options	one BEAUti options object, as returned by create_beauti_options
input_filenames	One or more FASTA filenames. Use get_fasta_filename to obtain a testing FASTA filename.
site_models	one or more site models, as returned by create_site_model
clock_models	a list of one or more clock models, as returned by create_clock_model
tree_priors	one or more tree priors, as returned by create_tree_prior
mrca_priors	a list of one or more Most Recent Common Ancestor priors, as returned by create_mrca_prior
posterior_crown_age	deprecated

Value

a character vector of XML strings

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_beast2_input_file to also save it to file. Use create_beast2_input_file_from_model to use an inference model as an input argument.

create_beast2_input_file shows more examples

Examples

```
text <- create_beast2_input(
  input_filename = get_fasta_filename()
)
testit::assert(substr(text[1], 1, 5) == "<?xml")
text[1]
testit::assert(tail(text, n = 1) == "</beast>")
```

`create_beast2_input_distr_lh` *Creates the likelihood section in the distribution section of a BEAST2 XML parameter file*

Description

Creates the likelihood section in the distribution section of a BEAST2 XML parameter file

Usage

```
create_beast2_input_distr_lh(site_models, clock_models, mrca_priors = NA,
  tipdates_filename = NA)
```

Arguments

`site_models`	one or more site models, as returned by `create_site_model`
`clock_models`	a list of one or more clock models, as returned by `create_clock_model`
`mrca_priors`	a list of one or more Most Recent Common Ancestor priors, as returned by `create_mrca_prior`
`tipdates_filename`	name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.

Note

this function is not intended for regular use, thus its long name length is accepted

Author(s)

Richèl J.C. Bilderbeek

See Also

this function is called by `create_beast2_input_distr`, together with `create_beast2_input_distr_prior`

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#     </distribution>
#     <distribution id="likelihood" ...>
#       HERE, where the ID of the distribution is 'likelihood'
#     </distribution>
# </distribution>
```

`create_beast2_input_distr_prior` *Creates the prior section in the distribution section of a BEAST2 XML parameter file*

Description

Creates the prior section in the distribution section of a BEAST2 XML parameter file

Usage

```
create_beast2_input_distr_prior(site_models, clock_models, tree_priors,
  mrca_priors = NA, tipdates_filename = NA)
```

Arguments

`site_models`	one or more site models, as returned by `create_site_model`
`clock_models`	a list of one or more clock models, as returned by `create_clock_model`
`tree_priors`	one or more tree priors, as returned by `create_tree_prior`
`mrca_priors`	a list of one or more Most Recent Common Ancestor priors, as returned by `create_mrca_prior`
`tipdates_filename`	name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.

Note

this function is not intended for regular use, thus its long name length is accepted

Author(s)

Richèl J.C. Bilderbeek

See Also

this function is called by `create_beast2_input_distr`, together with `create_beast2_input_distr_lh`

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#       HERE, where the ID of the distribution is 'prior'
#     </distribution>
#     <distribution id="likelihood" ...>
#     </distribution>
# </distribution>
```

`create_beast2_input_file` *Create a BEAST2 input file*

Description

Create a BEAST2 input file

Usage

```
create_beast2_input_file(input_filename, output_filename,
  site_model = create_jc69_site_model(),
  clock_model = create_strict_clock_model(),
  tree_prior = create_yule_tree_prior(), mrca_prior = NA,
  mcmc = create_mcmc(), beauti_options = create_beauti_options(),
  tipdates_filename = NA, input_filenames = "deprecated",
  site_models = "deprecated", clock_models = "deprecated",
  tree_priors = "deprecated", mrca_priors = "deprecated",
  posterior_crown_age = "deprecated")
```

Arguments

`input_filename`	A FASTA filename. Use `get_fasta_filename` to obtain a testing FASTA filename.
`output_filename`	Name of the XML parameter file created by this function. BEAST2 uses this file as input.
`site_model`	a site model, as returned by `create_site_model`
`clock_model`	a clock model, as returned by `create_clock_model`
`tree_prior`	a tree priors, as returned by `create_tree_prior`
`mrca_prior`	a Most Recent Common Ancestor prior, as returned by `create_mrca_prior`
`mcmc`	one MCMC as returned by `create_mcmc`
`beauti_options`	one BEAUti options object, as returned by `create_beauti_options`
`tipdates_filename`	name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.
`input_filenames`	One or more FASTA filenames. Use `get_fasta_filename` to obtain a testing FASTA filename.
`site_models`	one or more site models, as returned by `create_site_model`
`clock_models`	a list of one or more clock models, as returned by `create_clock_model`
`tree_priors`	one or more tree priors, as returned by `create_tree_prior`

mrca_priors a list of one or more Most Recent Common Ancestor priors, as returned by create_mrca_prior

posterior_crown_age deprecated

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_beast2_input_file_from_model to do the same with an inference model. See create_site_model for examples with different site models. See create_clock_model for examples with clock models. See create_tree_prior for examples with different tree priors. See create_mcmc for examples with a different MCMC setup.

Examples

```
# Get an example FASTA file
input_filename <- get_fasta_filename()

# The file created by beautier, a BEAST2 input file
output_filename <- "create_beast2_input_file.xml"

create_beast2_input_file(
  input_filename,
  output_filename
)
testthat::expect_true(file.exists(output_filename))
```

---

create_beast2_input_file_from_model

Create a BEAST2 input file from an inference model

Description

Create a BEAST2 input file from an inference model

Usage

```
create_beast2_input_file_from_model(input_filename, output_filename,
  inference_model = create_inference_model())
```

Arguments

input_filename	A FASTA filename. Use get_fasta_filename to obtain a testing FASTA filename.
output_filename	Name of the XML parameter file created by this function. BEAST2 uses this file as input.
inference_model	an Bayesian phylogenetic inference model, as can be created by create_inference_model. An inference model is the complete model setup in which a site model, clock model, tree prior and more are specified.

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

See create_site_model for examples with different site models. See create_clock_model for examples with clock models. See create_tree_prior for examples with different tree priors. See create_mcmc for examples with a different MCMC setup. Use create_beast2_input_file to do the same with the elements of an inference model.

Examples

```
# Get an example FASTA file
input_filename <- get_fasta_filename()

# The file created by beautier, a BEAST2 input file
output_filename <- tempfile()

create_beast2_input_file_from_model(
  input_filename,
  output_filename
)
testthat::expect_true(file.exists(output_filename))
```

---

create_beast2_input_screenlog

Creates the screenlog *section of the* logger *section of a BEAST2 XML parameter file*

Description

Creates the screenlog section of the logger section of a BEAST2 XML parameter file

Usage

```
create_beast2_input_screenlog()
```

Value

the XML text

Author(s)

Richèl J.C. Bilderbeek

---

create_beast2_input_tracelog

Creates the `tracelog` *section of the* `logger` *section of a BEAST2 XML parameter file*

Description

Creates the `tracelog` section of the `logger` section of a BEAST2 XML parameter file

Usage

```
create_beast2_input_tracelog(ids,
  site_models = list(create_jc69_site_model(id = ids)),
  clock_models = list(create_strict_clock_model(id = ids)),
  tree_priors = list(create_yule_tree_prior(id = ids)),
  mcmc = create_mcmc(), mrca_priors = NA, tipdates_filename = NA)
```

Arguments

`ids`	one or more alignments' IDs. IDs can be extracted from their FASTA filenames with `get_alignment_ids`)
`site_models`	one or more site models, as returned by `create_site_model`
`clock_models`	a list of one or more clock models, as returned by `create_clock_model`
`tree_priors`	one or more tree priors, as returned by `create_tree_prior`
`mcmc`	one MCMC as returned by `create_mcmc`
`mrca_priors`	a list of one or more Most Recent Common Ancestor priors, as returned by `create_mrca_prior`
`tipdates_filename`	name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.

Author(s)

Richèl J.C. Bilderbeek

Examples

```
created <- beautier:::create_beast2_input_tracelog(ids = 1)
expected <- c(
  paste0(
    "<logger id=\"tracelog\" fileName=\"1.log\" ",
    "logEvery=\"1000\" model=\"@posterior\" sanitiseHeaders=\"true\" ",
    "sort=\"smart\">"
  ),
  "    <log idref=\"posterior\"/>",
  "    <log idref=\"likelihood\"/>",
  "    <log idref=\"prior\"/>",
  "    <log idref=\"treeLikelihood.1\"/>",
  paste0(
    "    <log id=\"TreeHeight.t:1\" ",
    "spec=\"beast.evolution.tree.TreeHeightLogger\" ",
    "tree=\"@Tree.t:1\"/>"
  ),
  "    <log idref=\"YuleModel.t:1\"/>",
  "    <log idref=\"birthRate.t:1\"/>",
  "</logger>"
 )
 testthat::expect_equal(created, expected)
```

---

create_beast2_input_treelogs

Creates the `tracelog` *section of the* `logger` *section of a BEAST2 XML parameter file*

Description

Creates the `tracelog` section of the `logger` section of a BEAST2 XML parameter file

Usage

```
create_beast2_input_treelogs(clock_models)
```

Arguments

`clock_models`	a list of one or more clock models, as returned by `create_clock_model`

Author(s)

Richèl J.C. Bilderbeek

`create_beauti_options` *Function to create a set of BEAUti options.*

Description

BEAUti options are settings that differ between BEAUti version. The use of these options is mostly for testing older versions Whatever option chosen here, the created XML file will be valid.

Usage

```
create_beauti_options(capitalize_first_char_id = FALSE,
  nucleotides_uppercase = FALSE, beast2_version = "2.4",
  required = "", sequence_indent = 20)
```

Arguments

`capitalize_first_char_id`	must the ID of alignment start with a capital? TRUE if yes, FALSE if it can be left lower case (if it is lowercase)
`nucleotides_uppercase`	must the nucleotides of the DNA sequence be in uppercase?
`beast2_version`	the BEAST2 version
`required`	things that may be required, for example `BEAST v2.5.0`
`sequence_indent`	the number of spaces the XML `sequence` lines are indented

Value

a BEAUti options structure

Author(s)

Richèl J.C. Bilderbeek

Examples

```
beauti_options <- create_beauti_options(
  nucleotides_uppercase = TRUE,
  beast2_version = "2.5"
)
xml <- create_beast2_input(
  get_fasta_filename(),
  beauti_options = beauti_options
)
testit::assert(is.character(xml))
testit::assert(length(xml) > 1)
```

`create_beta_distr` *Create a beta distribution*

Description

Create a beta distribution

Usage

```
create_beta_distr(id = NA, alpha = 0, beta = 1)
```

Arguments

`id`	the distribution's ID
`alpha`	the alpha shape parameter, a numeric value. The value of alpha must be at least 0.0. For advanced usage, use the structure as returned by `create_alpha_param`.
`beta`	the beta shape parameter, a numeric value. The value of beta must be at least 1.0. For advanced usage, use the structure as returned by `create_beta_param`.

Value

a beta distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_distr` shows an overview of all supported distributions

Examples

```
beta_distr <- create_beta_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = beta_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_beta_param` *Create a parameter called beta*

Description

Create a parameter called beta

Usage

```
create_beta_param(id = NA, value = 1)
```

Arguments

`id`	the parameter's ID
`value`	value of the parameter

Value

a parameter called beta

Note

this parameter is used in a beta distribution (as returned by `create_beta_distr`) and gamma distribution (as returned by `create_gamma_distr`) and inverse-gamma distribution (as returned by `create_inv_gamma_distr`). It cannot be estimated (as a hyper parameter) yet.

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create the parameter
beta_param <- create_beta_param()

# Use the parameter in a distribution
gamma_distr <- create_gamma_distr(
  beta = beta_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
```

```
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = gamma_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_cbs_tree_prior` *Create a Coalescent Bayesian Skyline tree prior*

Description

Create a Coalescent Bayesian Skyline tree prior

Usage

```
create_cbs_tree_prior(id = NA, group_sizes_dimension = 5)
```

Arguments

`id`	an alignment's IDs. An ID can be extracted from its FASTA filename with `get_alignment_ids`)
`group_sizes_dimension`	the group sizes' dimension, as used by the CBS tree prior (see `create_cbs_tree_prior`)

Value

a Coalescent Bayesian Skyline tree_prior

Author(s)

Richèl J.C. Bilderbeek

See Also

An alignment ID can be extracted from its FASTA filename using `get_alignment_id`

Examples

```
cbs_tree_prior <- create_cbs_tree_prior()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_beautier_path("test_output_6.fas"),
  beast2_input_file,
  tree_prior = cbs_tree_prior
)
testit::assert(file.exists(beast2_input_file))
```

`create_ccp_tree_prior` *Create a Coalescent Constant Population tree prior*

Description

Create a Coalescent Constant Population tree prior

Usage

```
create_ccp_tree_prior(id = NA,
  pop_size_distr = beautier::create_one_div_x_distr())
```

Arguments

`id`	the ID of the alignment
`pop_size_distr`	the population distribution, as created by a `create_distr` function

Value

a Coalescent Constant Population tree_prior

Author(s)

Richèl J.C. Bilderbeek

See Also

An alignment ID can be extracted from its FASTA filename using `get_alignment_id`

Examples

```
ccp_tree_prior <- create_ccp_tree_prior()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = ccp_tree_prior
)
testit::assert(file.exists(beast2_input_file))
```

`create_cep_tree_prior` *Create a Coalescent Exponential Population tree prior*

Description

Create a Coalescent Exponential Population tree prior

Usage

```
create_cep_tree_prior(id = NA,
  pop_size_distr = create_one_div_x_distr(),
  growth_rate_distr = create_laplace_distr())
```

Arguments

`id`	the ID of the alignment
`pop_size_distr`	the population distribution, as created by a `create_distr` function
`growth_rate_distr`	the growth rate distribution, as created by a `create_distr` function

Value

a Coalescent Exponential Population tree_prior

Author(s)

Richèl J.C. Bilderbeek

See Also

An alignment ID can be extracted from its FASTA filename using `get_alignment_id`

Examples

```
cep_tree_prior <- create_cep_tree_prior()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = cep_tree_prior
)
testit::assert(file.exists(beast2_input_file))
```

`create_clock_model` *General function to create a clock model*

Description

General function to create a clock model

Usage

```
create_clock_model(name, id, ...)
```

Arguments

`name`	the clock model name. Valid names can be found in `get_clock_model_names`
`id`	a clock model's ID
`...`	specific clock model parameters

Value

a valid clock model

Note

Prefer using the named function `create_rln_clock_model` and `create_strict_clock_model`

Author(s)

Richèl J.C. Bilderbeek

See Also

An alignment ID can be extracted from its FASTA filename using `get_alignment_id`. For more examples about creating a relaxed log-normal clock model, see `create_rln_clock_model`. For more examples about creating a strict clock model, see `create_strict_clock_model`.

Examples

```
rln_clock_model <- create_rln_clock_model()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  clock_model = rln_clock_model
)
testit::assert(file.exists(beast2_input_file))

strict_clock_model <- create_strict_clock_model()
```

```
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  clock_model = strict_clock_model
)
testit::assert(file.exists(beast2_input_file))
```

`create_clock_models` *Creates all supported clock models, which is a list of the types returned by* `create_rln_clock_model`*, and* `create_strict_clock_model`

Description

Creates all supported clock models, which is a list of the types returned by `create_rln_clock_model`, and `create_strict_clock_model`

Usage

```
create_clock_models()
```

Value

a list of site_models

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_model to create a clock model

Examples

```
clock_models <- create_clock_models()
testit::assert(beautier:::is_rln_clock_model(clock_models[[1]]))
testit::assert(beautier:::is_strict_clock_model(clock_models[[2]]))
```

`create_clock_models_from_names` *Create clock models from their names*

Description

Create clock models from their names

Usage

```
create_clock_models_from_names(clock_model_names)
```

Arguments

`clock_model_names`	one or more names of a clock model, must be name among those returned by `get_clock_model_names`

Value

a list of one or more clock models

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_models to get all clock models

Examples

```
names <- get_clock_model_names()
clock_models <- create_clock_models_from_names(names)

for (i in seq_along(names)) {
  testthat::expect_equal(names[i], clock_models[[i]]$name)
}
```

`create_clock_model_from_name` *Create a clock model from name*

Description

Create a clock model from name

Usage

```
create_clock_model_from_name(clock_model_name)
```

Arguments

`clock_model_name`	name of a clock model, must be a name as returned by `get_clock_model_names`

Value

a clock model, as can be created by create_clock_model

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `create_clock_model` to create a clock model

Examples

```
clock_model_names <- get_clock_model_names()
for (clock_model_name in clock_model_names) {
  clock_model <- create_clock_model_from_name(clock_model_name)
  testthat::expect_equal(clock_model_name, clock_model$name)
}
```

create_clock_rate_param *Create a parameter called* clock_rate*, as needed by* create_strict_clock_model

Description

Create a parameter called clock_rate, as needed by create_strict_clock_model

Usage

```
create_clock_rate_param(value = "1.0", id = NA)
```

Arguments

value	value of the parameter
id	the parameter's ID

Value

a parameter called rate

Note

It cannot be estimated (as a hyper parameter) yet.

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_param contains a list of all parameters that can be created

Examples

```
clock_rate_param <- create_clock_rate_param(
  id = "anthus_aco", value = 1.0
)

# Use the parameter in a clock model
strict_clock_model <- create_strict_clock_model(
  clock_rate_param = clock_rate_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
```

```
  beast2_input_file,
  clock_model = strict_clock_model
)
testit::assert(file.exists(beast2_input_file))
```

`create_distr` *General function to create a distribution.*

Description

General function to create a distribution.

Usage

```
create_distr(name, id, ...)
```

Arguments

`name`	the distribution name. Valid names can be found in `get_distr_names`
`id`	the distribution's ID
`...`	specific distribution parameters

Value

a distribution

Note

Prefer using the named functions `create_beta_distr`, `create_exp_distr`, `create_gamma_distr`, `create_inv_gamma_distr`, `create_laplace_distr`, `create_log_normal_distr`, `create_normal_distr`, `create_one_div_x_distr`, `create_poisson_distr` and `create_uniform_distr`

See `create_beta_distr`, `create_exp_distr`, `create_gamma_distr`, `create_inv_gamma_distr`, `create_laplace_distr`, `create_log_normal_distr`, `create_normal_distr`, `create_one_div_x_distr`, `create_poisson_distr` and `create_uniform_distr` for examples how to use those distributions

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# Use any distribution
distr <- create_beta_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
```

```
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_exp_distr` *Create an exponential distribution*

Description

Create an exponential distribution

Usage

```
create_exp_distr(id = NA, mean = 1)
```

Arguments

`id`	the distribution's ID
`mean`	the mean parameter, a numeric value. For advanced usage, use the structure as returned by `create_mean_param`

Value

an exponential distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_distr` shows an overview of all supported distributions

Examples

```
exp_distr <- create_exp_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = exp_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_gamma_distr` *Create a gamma distribution*

Description

Create a gamma distribution

Usage

```
create_gamma_distr(id = NA, alpha = 0.5396, beta = 0.3819)
```

Arguments

`id`	the distribution's ID
`alpha`	the alpha shape parameter, a numeric value. For advanced usage, use the structure as returned by `create_alpha_param`
`beta`	the beta shape parameter, a numeric value. For advanced usage, use the structure as returned by `create_beta_param`

Value

a gamma distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_distr` shows an overview of all supported distributions

Examples

```
gamma_distr <- create_gamma_distr(
   alpha = 0.05,
   beta = 10.0
)

gtr_site_model <- create_gtr_site_model(
  rate_ac_prior_distr = gamma_distr
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = gtr_site_model
)
testit::assert(file.exists(beast2_input_file))
```

create_gamma_site_model *Create a gamma site model, part of a site model*

Description

Create a gamma site model, part of a site model

Usage

```
create_gamma_site_model(gamma_cat_count = "0", gamma_shape = "1.0",
  prop_invariant = "0.0", gamma_shape_prior_distr = NA,
  freq_equilibrium = "estimated")
```

Arguments

gamma_cat_count	the number of gamma categories, must be an integer with value zero or more
gamma_shape	gamma curve shape parameter
prop_invariant	the proportion invariant, must be a value from 0.0 to 1.0
gamma_shape_prior_distr	the distribution of the gamma shape prior. gamma_shape_prior_distr must be NA for a gamma_cat_count of zero or one. For a gamma_cat_count of two or more, leaving gamma_shape_prior_distr equal to its default value of NA, a default distribution is used. Else gamma_shape_prior_distr must be a distribution, as can be created by create_distr
freq_equilibrium	the frequency in which the rates are at equilibrium are either estimated, empirical or all_equal. get_freq_equilibrium_names returns the possible values for freq_equilibrium

Value

a gamma site model

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_gamma_site_model to create a gamma site model

Examples

```
gamma_site_model <- create_gamma_site_model(prop_invariant = 0.5)

site_model <- create_hky_site_model(gamma_site_model = gamma_site_model)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  site_model = site_model
)
testit::assert(file.exists(beast2_input_file))
```

`create_gtr_site_model` *Create a GTR site model*

Description

Create a GTR site model

Usage

```
create_gtr_site_model(id = NA,
  gamma_site_model = create_gamma_site_model(),
  rate_ac_prior_distr = create_gamma_distr(alpha = 0.05, beta =
  create_beta_param(value = "10.0")),
  rate_ag_prior_distr = create_gamma_distr(alpha = 0.05, beta =
  create_beta_param(value = "20.0")),
  rate_at_prior_distr = create_gamma_distr(alpha = 0.05, beta =
  create_beta_param(value = "10.0")),
  rate_cg_prior_distr = create_gamma_distr(alpha = 0.05, beta =
  create_beta_param(value = "10.0")),
  rate_gt_prior_distr = create_gamma_distr(alpha = 0.05, beta =
  create_beta_param(value = "10.0")),
  rate_ac_param = create_rate_ac_param(),
  rate_ag_param = create_rate_ag_param(),
  rate_at_param = create_rate_at_param(),
  rate_cg_param = create_rate_cg_param(),
  rate_ct_param = create_rate_ct_param(),
  rate_gt_param = create_rate_gt_param(),
  freq_equilibrium = "estimated")
```

Arguments

`id`	the IDs of the alignment (can be extracted from the FASTA filename using `get_alignment_id`)
`gamma_site_model`	a gamma site model, as created by `create_gamma_site_model`

rate_ac_prior_distr
: the AC rate prior distribution, as returned by create_distr

rate_ag_prior_distr
: the AG rate prior distribution, as returned by create_distr

rate_at_prior_distr
: the AT rate prior distribution, as returned by create_distr

rate_cg_prior_distr
: the CG rate prior distribution, as returned by create_distr

rate_gt_prior_distr
: the GT rate prior distribution, as returned by create_distr

rate_ac_param
: the 'rate AC' parameter, a numeric value. For advanced usage, use the structure as returned by create_rate_ac_param

rate_ag_param
: the 'rate AG' parameter, a numeric value. For advanced usage, use the structure as returned by create_rate_ag_param

rate_at_param
: the 'rate AT' parameter, a numeric value. For advanced usage, use the structure as returned by create_rate_at_param

rate_cg_param
: the 'rate CG' parameter, a numeric value. For advanced usage, use the structure as returned by create_rate_cg_param

rate_ct_param
: the 'rate CT' parameter, a numeric value. For advanced usage, use the structure as returned by create_rate_ct_param

rate_gt_param
: the 'rate GT' parameter, a numeric value. For advanced usage, use the structure as returned by create_rate_gt_param

freq_equilibrium
: the frequency in which the rates are at equilibrium are either estimated, empirical or all_equal. get_freq_equilibrium_names returns the possible values for freq_equilibrium

Value

a GTR site_model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
gtr_site_model <- create_gtr_site_model(
  rate_ac_param = 1.2,
  rate_ag_param = 2.3,
  rate_at_param = 3.4,
  rate_cg_param = 4.5,
  rate_ct_param = 5.6,
  rate_gt_param = 6.7
)

beast2_input_file <- tempfile(fileext = ".xml")
```

```
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = gtr_site_model
)
testit::assert(file.exists(beast2_input_file))
```

`create_hky_site_model` *Create an HKY site model*

Description

Create an HKY site model

Usage

```
create_hky_site_model(id = NA, kappa = "2.0",
  gamma_site_model = create_gamma_site_model(),
  kappa_prior_distr = create_log_normal_distr(m = create_m_param(value =
  "1.0"), s = 1.25), freq_equilibrium = "estimated")
```

Arguments

`id`	the IDs of the alignment (can be extracted from the FASTA filename using `get_alignment_id`)
`kappa`	the kappa
`gamma_site_model`	a gamma site model, as created by `create_gamma_site_model`
`kappa_prior_distr`	the distribution of the kappa prior, which is a log-normal distribution (as created by `create_log_normal_distr`) by default
`freq_equilibrium`	the frequency in which the rates are at equilibrium are either `estimated`, `empirical` or `all_equal`. `get_freq_equilibrium_names` returns the possible values for `freq_equilibrium`

Value

an HKY site_model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
hky_site_model <- create_hky_site_model()

create_beast2_input_file(
  input_filename = get_fasta_filename(),
  "beast.xml",
  site_model = hky_site_model
)
```

`create_inference_model` *Create a Bayesian phylogenetic inference model.*

Description

Create a Bayesian phylogenetic inference model, as can be done by BEAUti.

Usage

```
create_inference_model(site_model = create_jc69_site_model(),
  clock_model = create_strict_clock_model(),
  tree_prior = create_yule_tree_prior(), mrca_prior = NA,
  mcmc = create_mcmc(), beauti_options = create_beauti_options(),
  tipdates_filename = NA)
```

Arguments

`site_model`	a site model, as returned by `create_site_model`
`clock_model`	a clock model, as returned by `create_clock_model`
`tree_prior`	a tree priors, as returned by `create_tree_prior`
`mrca_prior`	a Most Recent Common Ancestor prior, as returned by `create_mrca_prior`
`mcmc`	one MCMC as returned by `create_mcmc`
`beauti_options`	one BEAUti options object, as returned by `create_beauti_options`
`tipdates_filename`	name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.

Value

an MCMC configuration

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_inference_model to create an inference model

Examples

```
# Create an MCMC chain with 50 states
inference_model <- create_inference_model(
  mcmc = create_mcmc(chain_length = 50000, store_every = 1000)
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file_from_model(
  get_fasta_filename(),
  beast2_input_file,
  inference_model = inference_model
)
testit::assert(file.exists(beast2_input_file))
```

---

create_inv_gamma_distr

Create an inverse-gamma distribution

Description

Create an inverse-gamma distribution

Usage

```
create_inv_gamma_distr(id = NA, alpha = 0, beta = 1)
```

Arguments

id	the distribution's ID
alpha	the alpha shape parameter, a numeric value. For advanced usage, use the structure as returned by create_alpha_param
beta	the beta shape parameter, a numeric value. For advanced usage, use the structure as returned by create_beta_param

Value

an inverse-gamma distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_distr shows an overview of all supported distributions

Examples

```
inv_gamma_distr <- create_inv_gamma_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = inv_gamma_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_jc69_site_model

Create a JC69 site model

Description

Create a JC69 site model

Usage

```
create_jc69_site_model(id = NA,
  gamma_site_model = create_gamma_site_model())
```

Arguments

id	the IDs of the alignment (can be extracted from the FASTA filename using get_alignment_id)
gamma_site_model	a gamma site model, as created by create_gamma_site_model

Value

a JC69 site_model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
jc69_site_model <- create_jc69_site_model()

create_beast2_input_file(
  input_filename = get_fasta_filename(),
  "beast.xml",
  site_model = jc69_site_model
)
```

`create_kappa_1_param` *Create a parameter called kappa 1*

Description

Create a parameter called kappa 1

Usage

```
create_kappa_1_param(id = NA, lower = "0.0", value = "2.0",
  estimate = TRUE)
```

Arguments

`id`	the parameter's ID
`lower`	lowest possible value of the parameter. If the parameter is estimated, `lower` must be less than `value`
`value`	value of the parameter
`estimate`	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise

Value

a parameter called kappa 1

Author(s)

Richèl J.C. Bilderbeek

create_kappa_2_param *Create a parameter called kappa 2*

Description

Create a parameter called kappa 2

Usage

```
create_kappa_2_param(id = NA, lower = "0.0", value = "2.0",
  estimate = TRUE)
```

Arguments

id	the parameter's ID
lower	lowest possible value of the parameter. If the parameter is estimated, lower must be less than value
value	value of the parameter
estimate	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise

Value

a parameter called kappa 2

Author(s)

Richèl J.C. Bilderbeek

create_lambda_param *Create a parameter called lambda*

Description

Create a parameter called lambda

Usage

```
create_lambda_param(id = NA, value = 0)
```

Arguments

id	the parameter's ID
value	value of the parameter

Value

a parameter called lambda

Note

this parameter is used in a Poisson distribution (as returned by `create_poisson_distr`)

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create the parameter
lambda_param <- create_lambda_param()

# Use the parameter in a distribution
poisson_distr <- create_poisson_distr(
  lambda = lambda_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = poisson_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_laplace_distr *Create a Laplace distribution*

Description

Create a Laplace distribution

Usage

```
create_laplace_distr(id = NA, mu = 0, scale = 1)
```

Arguments

id	the distribution's ID
mu	the mu parameter, a numeric value. For advanced usage, use the structure as returned by create_mu_param
scale	the scale parameter, a numeric value. For advanced usage, use the structure as returned by create_scale_param

Value

a Laplace distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_distr shows an overview of all supported distributions

Examples

```
laplace_distr <- create_laplace_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = laplace_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_log_normal_distr

Create a log-normal distribution

Description

Create a log-normal distribution

Usage

```
create_log_normal_distr(id = NA, m = 0, s = 0)
```

Arguments

id	the distribution's ID
m	the m parameter, a numeric value. For advanced usage, use the structure as returned by create_m_param
s	the s parameter, a numeric value. For advanced usage, use the structure as returned by create_s_param

Value

a log-normal distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_distr shows an overview of all supported distributions

Examples

```
log_normal_distr <- create_log_normal_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = log_normal_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_mcmc *Create an MCMC configuration.*

Description

Create an MCMC configuration, as in the BEAUti MCMC tab. The number of states that will be saved equals the chain length (chain_length) divided by the number of states between each sampling event (store_every)

Usage

```
create_mcmc(chain_length = 1e+07, store_every = -1)
```

Arguments

chain_length	length of the MCMC chain
store_every	number of states the MCMC will process before the posterior's state will be saved to file. Use -1 or NA to use the default frequency.

Value

an MCMC configuration

Author(s)

Richèl J.C. Bilderbeek

See Also

- are_equal_mcmcs to check if two MCMCs are equal

Examples

```
# Create an MCMC chain with 50 states
mcmc <- create_mcmc(chain_length = 50000, store_every = 1000)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  mcmc = mcmc
)
testit::assert(file.exists(beast2_input_file))
```

---

create_mean_param *Create a parameter called mean*

Description

Create a parameter called mean

Usage

```
create_mean_param(id = NA, value = 0)
```

Arguments

id	the parameter's ID
value	value of the parameter

Value

a parameter called mean

Note

this parameter is used in an exponential distribution (as returned by `create_exp_distr`) and normal distribution (as returned by `create_normal_distr`). It cannot be estimated (as a hyper parameter) yet.

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create the parameter
mean_param <- create_mean_param(value = 1.0)

# Use the parameter in a distribution
exp_distr <- create_exp_distr(
  mean = mean_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = exp_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_mrca_prior *Create a Most Recent Common Ancestor prior*

Description

Create a Most Recent Common Ancestor prior

Usage

```
create_mrca_prior(alignment_id = NA, taxa_names = NA,
  is_monophyletic = FALSE, mrca_distr = NA, name = NA,
  clock_prior_distr_id = NA)
```

Arguments

alignment_id	ID of the alignment, as returned by get_alignment_id. Keep at NA to have it initialized automatically
taxa_names	names of the taxa, as returned by get_taxa_names. Keep at NA to have it initialized automatically, using all taxa in the alignment
is_monophyletic	boolean to indicate monophyly is assumed in a Most Recent Common Ancestor prior, as returned by create_mrca_prior
mrca_distr	the distribution used by the MRCA prior. Can be NA (the default) or any distribution returned by create_distr
name	the unique name of the MRCA prior, for example a genus, family, order or even class name. Leave at NA to have it named automatically
clock_prior_distr_id	ID of an MRCA clock model's distribution. Keep at NA to have it initialized automatically

Value

an MRCA prior

Author(s)

Richèl J.C. Bilderbeek

Examples

```
fasta_filename <- get_beautier_path("anthus_aco.fas")

# The first two taxa are sister species
mrca_prior <- create_mrca_prior(
  alignment_id = get_alignment_id(fasta_filename = fasta_filename),
  taxa_names = get_taxa_names(filename = fasta_filename)[1:2]
)

# Set the crown age
mrca_prior <- create_mrca_prior(
  alignment_id = get_alignment_id(fasta_filename = fasta_filename),
  taxa_names = get_taxa_names(filename = fasta_filename),
  is_monophyletic = TRUE
)
```

`create_mu_param` *Create a parameter called mu*

Description

Create a parameter called mu

Usage

```
create_mu_param(id = NA, value = 0)
```

Arguments

`id`	the parameter's ID
`value`	value of the parameter

Value

a parameter called mu

Note

this parameter is used in a Laplace distribution (as returned by `create_laplace_distr`). It cannot be estimated (as a hyper parameter) yet.

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create the parameter
mu_param <- create_mu_param()

# Use the parameter in a distribution
laplace_distr <- create_laplace_distr(
  mu = mu_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
```

```
    birth_rate_distr = laplace_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_m_param` *Create a parameter called m*

Description

Create a parameter called m

Usage

```
create_m_param(id = NA, value = 0)
```

Arguments

`id`	the parameter's ID
`value`	value of the parameter

Value

a parameter called m

Note

this parameter is used in a log-normal distribution (as returned by `create_log_normal_distr`) It cannot be estimated (as a hyper parameter) yet.

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create the parameter
m_param <- create_m_param()

# Use the parameter in a distribution
log_normal_distr <- create_log_normal_distr(
  m = m_param
)

# Use the distribution to create a BEAST2 input file
```

```
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = log_normal_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_nested_sampling_mcmc` *Create an MCMC object to estimate the marginal likelihood using Nested Sampling.*

Description

This will result in a BEAST run that estimates the marginal likelihood until convergence is achieved. In this context, `chain_length` is only an upper bound to the length of that run.

Usage

```
create_nested_sampling_mcmc(chain_length = 1e+07, store_every = -1,
  particle_count = 1, sub_chain_length = 5000, epsilon = "1e-12")
```

Arguments

`chain_length`	upper bound to the length of the MCMC chain
`store_every`	number of states the MCMC will process before the posterior's state will be saved to file. Use -1 or `NA` to use the default frequency.
`particle_count`	number of particles
`sub_chain_length`	sub-chain length
`epsilon`	epsilon

Value

an MCMC object

Author(s)

Richèl J.C. Bilderbeek

References

* [1] Patricio Maturana Russel, Brendon J Brewer, Steffen Klaere, Remco R Bouckaert; Model Selection and Parameter Inference in Phylogenetics Using Nested Sampling, Systematic Biology, 2018, syy050, https://doi.org/10.1093/sysbio/syy050

See Also

Use create_nested_sampling_mcmc to create a nested sampling MCMC

Examples

```
mcmc <- create_mcmc_nested_sampling(
  chain_length = 1e7,
  store_every = 1000,
  particle_count = 1,
  sub_chain_length = 1000,
  epsilon = 1e-12
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  mcmc = mcmc
)
testit::assert(file.exists(beast2_input_file))
```

---

create_normal_distr *Create an normal distribution*

Description

Create an normal distribution

Usage

```
create_normal_distr(id = NA, mean = 0, sigma = 1)
```

Arguments

id	the distribution's ID
mean	the mean parameter, a numeric value. For advanced usage, use the structure as returned by create_mean_param
sigma	the sigma parameter, a numeric value. For advanced usage, use the structure as returned by create_sigma_param

Value

a normal distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_distr shows an overview of all supported distributions

Examples

```
normal_distr <- create_normal_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = normal_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_one_div_x_distr

Create a 1/x distribution

Description

Create a 1/x distribution

Usage

```
create_one_div_x_distr(id = NA)
```

Arguments

id	the distribution's ID

Value

a 1/x distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_distr shows an overview of all supported distributions

Examples

```
one_div_x_distr <- create_one_div_x_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = one_div_x_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_param *General function to create a parameter.*

Description

General function to create a parameter.

Usage

```
create_param(name, id, value, ...)
```

Arguments

name	the parameters' name. Valid names can be found in get_param_names
id	the parameter's ID
value	value of the parameter
...	specific parameter parameters

Value

a parameter

Note

Prefer using the named functions create_alpha_param, create_beta_param, create_clock_rate_param, create_kappa_1_param, create_kappa_2_param, create_lambda_param, create_m_param, create_mean_param, create_mu_param, create_rate_ac_param, create_rate_ag_param, create_rate_at_param, create_rate_cg_param, create_rate_ct_param, create_rate_gt_param, create_s_param, create_scale_param, and create_sigma_param

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# Create an alpha parameter
alpha_param <- create_alpha_param()

# Use the parameter in a distribution
beta_distr <- create_beta_distr(
  alpha = alpha_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = beta_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_poisson_distr *Create a Poisson distribution*

Description

Create a Poisson distribution

Usage

```
create_poisson_distr(id = NA, lambda = 0)
```

Arguments

id	the distribution's ID
lambda	the lambda parameter, a numeric value. For advanced usage, use the structure as returned by create_lambda_param

Value

a Poisson distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_distr shows an overview of all supported distributions

Examples

```
poisson_distr <- create_poisson_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = poisson_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_rate_ac_param` *Create a parameter called 'rate AC'*

Description

Create a parameter called 'rate AC'

Usage

```
create_rate_ac_param(id = NA, estimate = TRUE, value = "1.0",
  lower = "0.0")
```

Arguments

`id`	the parameter's ID
`estimate`	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise
`value`	value of the parameter
`lower`	lowest possible value of the parameter. If the parameter is estimated, `lower` must be less than `value`

Value

a parameter called 'rate AC'

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create parameter
rate_ac_param <- create_rate_ac_param(value = 1, estimate = FALSE)

# Use the parameter to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = create_gtr_site_model(
    rate_ac_param = rate_ac_param
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_rate_ag_param` *Create a parameter called 'rate AG'*

Description

Create a parameter called 'rate AG'

Usage

```
create_rate_ag_param(id = NA, estimate = TRUE, value = "1.0",
  lower = "0.0")
```

Arguments

`id`	the parameter's ID
`estimate`	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise
`value`	value of the parameter
`lower`	lowest possible value of the parameter. If the parameter is estimated, `lower` must be less than `value`

Value

a parameter called 'rate AG'

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create parameter
rate_ag_param <- create_rate_ag_param(value = 1, estimate = FALSE)

# Use the parameter to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = create_gtr_site_model(
    rate_ag_param = rate_ag_param
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_rate_at_param` *Create a parameter called 'rate AT'*

Description

Create a parameter called 'rate AT'

Usage

```
create_rate_at_param(id = NA, estimate = TRUE, value = "1.0",
  lower = "0.0")
```

Arguments

`id`	the parameter's ID
`estimate`	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise
`value`	value of the parameter
`lower`	lowest possible value of the parameter. If the parameter is estimated, `lower` must be less than `value`

Value

a parameter called 'rate AT'

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create parameter
rate_at_param <- create_rate_at_param(value = 1, estimate = FALSE)

# Use the parameter to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = create_gtr_site_model(
    rate_at_param = rate_at_param
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_rate_cg_param *Create a parameter called 'rate CG'*

Description

Create a parameter called 'rate CG'

Usage

```
create_rate_cg_param(id = NA, estimate = TRUE, value = "1.0",
  lower = "0.0")
```

Arguments

id	the parameter's ID
estimate	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise
value	value of the parameter
lower	lowest possible value of the parameter. If the parameter is estimated, lower must be less than value

Value

a parameter called 'rate CG'

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_param contains a list of all parameters that can be created

Examples

```
# Create parameter
rate_cg_param <- create_rate_cg_param(value = 1, estimate = FALSE)

# Use the parameter to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = create_gtr_site_model(
    rate_cg_param = rate_cg_param
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_rate_ct_param` *Create a parameter called 'rate CT'*

Description

Create a parameter called 'rate CT'

Usage

```
create_rate_ct_param(id = NA, value = "1.0", lower = "0.0")
```

Arguments

`id`	the parameter's ID
`value`	value of the parameter
`lower`	lowest possible value of the parameter. If the parameter is estimated, `lower` must be less than `value`

Value

a parameter called 'rate CT'

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create parameter
rate_ct_param <- create_rate_ct_param(value = 1)

# Use the parameter to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = create_gtr_site_model(
    rate_ct_param = rate_ct_param
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_rate_gt_param *Create a parameter called 'rate GT'*

Description

Create a parameter called 'rate GT'

Usage

```
create_rate_gt_param(id = NA, estimate = TRUE, value = "1.0",
  lower = "0.0")
```

Arguments

id	the parameter's ID
estimate	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise
value	value of the parameter
lower	lowest possible value of the parameter. If the parameter is estimated, lower must be less than value

Value

a parameter called 'rate GT'

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_param contains a list of all parameters that can be created

Examples

```
# Create parameter
rate_gt_param <- create_rate_gt_param(value = 1, estimate = FALSE)

# Use the parameter to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  site_model = create_gtr_site_model(
    rate_gt_param = rate_gt_param
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_rln_clock_model`

Create a relaxed log-normal clock model

Description

Create a relaxed log-normal clock model

Usage

```
create_rln_clock_model(id = NA,
  mean_rate_prior_distr = create_uniform_distr(),
  ucldstdev_distr = create_gamma_distr(), mparam_id = NA,
  mean_clock_rate = "1.0", n_rate_categories = -1,
  normalize_mean_clock_rate = FALSE, dimension = NA)
```

Arguments

`id`	an alignment's IDs. An ID can be extracted from its FASTA filename with `get_alignment_ids`)
`mean_rate_prior_distr`	the mean clock rate prior distribution, as created by a `create_distr` function
`ucldstdev_distr`	the standard deviation of the uncorrelated log-normal distribution, as created by a `create_distr` function
`mparam_id`	the ID of the M parameter in the `branchRateModel`, set to NA to have it initialized
`mean_clock_rate`	the mean clock rate, 1.0 by default (is called `ucld_stdev` in XML, where `ucld_stdev` is always 0.1)
`n_rate_categories`	the number of rate categories. -1 is default, 0 denotes as much rates as branches

normalize_mean_clock_rate
: normalize the mean clock rate

dimension
: the dimensionality of the relaxed clock model. Leave NA to let beautier calculate it. Else, the dimensionality of the clock equals twice the number of taxa minus two.

Value

a relaxed log-normal clock_model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
rln_clock_model <- create_rln_clock_model()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  clock_model = rln_clock_model
)
testit::assert(file.exists(beast2_input_file))

rln_clock_model_exp <- create_rln_clock_model(
  mean_rate_prior_distr = create_exp_distr()
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  clock_model = rln_clock_model_exp
)
testit::assert(file.exists(beast2_input_file))
```

---

create_scale_param *Create a parameter called scale*

Description

Create a parameter called scale

Usage

```
create_scale_param(id = NA, value = 0)
```

Arguments

id	the parameter's ID
value	value of the parameter

Value

a parameter called scale

Note

this parameter is used in a Laplace distribution (as returned by create_laplace_distr)

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_param contains a list of all parameters that can be created

Examples

```
# Create the parameter
scale_param <- create_scale_param()

# Use the parameter in a distribution
laplace_distr <- create_laplace_distr(
  scale = scale_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = laplace_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_sigma_param *Create a parameter called sigma*

Description

Create a parameter called sigma

Usage

```
create_sigma_param(id = NA, value = 1)
```

Arguments

id	the parameter's ID
value	value of the parameter

Value

a parameter called sigma

Note

this parameter is used in a normal distribution (as returned by create_normal_distr)

Author(s)

Richèl J.C. Bilderbeek

See Also

the function create_param contains a list of all parameters that can be created

Examples

```
# Create the parameter
sigma_param <- create_sigma_param()

# Use the parameter in a distribution
normal_distr <- create_normal_distr(
  sigma = sigma_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = normal_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_site_model` *General function to create a site model.*

Description

General function to create a site model.

Usage

```
create_site_model(name, id, gamma_site_model = create_gamma_site_model(),
  ...)
```

Arguments

`name`	the site model name. Valid names can be found in `get_site_model_names`
`id`	the IDs of the alignment (can be extracted from the FASTA filename using `get_alignment_id`)
`gamma_site_model`	a gamma site model, as created by `create_gamma_site_model`
`...`	specific site model parameters

Value

a site_model

Note

Prefer using the named functions `create_gtr_site_model`, `create_hky_site_model`,, `create_jc69_site_model`, and `create_tn93_site_model`

Author(s)

Richèl J.C. Bilderbeek

See Also

See `create_gtr_site_model` for more examples with a GTR site model. See `create_hky_site_model` for more examples with an HKY site model. See `create_jc69_site_model` for more examples with a JC69 site model. See `create_tn93_site_model` for more examples with a TN93 site model

Examples

```
# GTR
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  output_filename = "example_gtr.xml",
  site_model = create_gtr_site_model()
)
```

```
testthat::expect_true(file.exists("example_gtr.xml"))

# HKY
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  output_filename = "example_hky.xml",
  site_model = create_hky_site_model()
)
testthat::expect_true(file.exists("example_hky.xml"))

# JC69
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  output_filename = "example_jc69.xml",
  site_model = create_jc69_site_model()
)
testthat::expect_true(file.exists("example_jc69.xml"))

# TN93
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  output_filename = "example_tn93.xml",
  site_model = create_tn93_site_model()
)
testthat::expect_true(file.exists("example_tn93.xml"))
```

`create_site_models` *Creates all supported site models which is a list of the types returned by* `create_gtr_site_model`, `create_hky_site_model`, `create_jc69_site_model` *and* `create_tn93_site_model`

Description

Creates all supported site models which is a list of the types returned by `create_gtr_site_model`, `create_hky_site_model`, `create_jc69_site_model` and `create_tn93_site_model`

Usage

```
create_site_models()
```

Value

a list of site_models

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_model to create a site model

Examples

```
# All created site models are a kind of site model
site_models <- beautier:::create_site_models()
testit::assert(beautier:::is_gtr_site_model(site_models[[1]]))
testit::assert(beautier:::is_hky_site_model(site_models[[2]]))
testit::assert(beautier:::is_jc69_site_model(site_models[[3]]))
testit::assert(beautier:::is_tn93_site_model(site_models[[4]]))

# Names are conformant
for (site_model in site_models) {
  testit::assert(site_model$name %in% get_site_model_names())
}
```

---

create_site_models_from_names

Create site models from their names

Description

Create site models from their names

Usage

```
create_site_models_from_names(site_model_names)
```

Arguments

site_model_names	one or more names of a site model, must be name among those returned by get_site_model_names

Value

one or more site models

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_model to create a site model

Examples

```
names <- get_site_model_names()
site_models <- create_site_models_from_names(names)

for (i in seq_along(names)) {
  testthat::expect_equal(names[i], site_models[[i]]$name)
}
```

`create_site_model_from_name` *Create a site model from name*

Description

Create a site model from name

Usage

```
create_site_model_from_name(site_model_name)
```

Arguments

`site_model_name`	name of a site model, must be a name as returned by `get_site_model_names`

Value

a site model

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_model to create a site model

Examples

```
site_model_names <- get_site_model_names()
for (site_model_name in site_model_names) {
  site_model <- create_site_model_from_name(site_model_name)
  testthat::expect_equal(site_model_name, site_model$name)
}
```

create_strict_clock_model *Create a strict clock model*

Description

Create a strict clock model

Usage

```
create_strict_clock_model(id = NA,
  clock_rate_param = create_clock_rate_param(),
  clock_rate_distr = create_uniform_distr())
```

Arguments

id	an alignment's IDs. An ID can be extracted from its FASTA filename with get_alignment_ids)
clock_rate_param	the clock rate's parameter, a numeric value. For advanced usage, use the structure as created by the create_clock_rate_param function
clock_rate_distr	the clock rate's distribution, as created by a create_distr function

Value

a strict clock_model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
strict_clock_model <- create_strict_clock_model(
  clock_rate_param = 1.0,
  clock_rate_distr = create_uniform_distr()
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  clock_model = strict_clock_model
)
testit::assert(file.exists(beast2_input_file))

strict_clock_model_gamma <- create_strict_clock_model(
```

```
  clock_rate_distr = create_gamma_distr()
)

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  get_fasta_filename(),
  beast2_input_file,
  clock_model = strict_clock_model_gamma
)
testit::assert(file.exists(beast2_input_file))
```

`create_s_param` *Create a parameter called s*

Description

Create a parameter called s

Usage

```
create_s_param(id = NA, value = 0, lower = 0, upper = Inf)
```

Arguments

`id`	the parameter's ID
`value`	value of the parameter
`lower`	lowest possible value of the parameter. If the parameter is estimated, `lower` must be less than `value`
`upper`	upper value of the parameter

Value

a parameter called s

Note

this parameter is used in a log-normal distribution (as returned by `create_log_normal_distr`)

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_param` contains a list of all parameters that can be created

Examples

```
# Create the parameter
s_param <- create_s_param()

# Use the parameter in a distribution
log_normal_distr <- create_log_normal_distr(
  s = s_param
)

# Use the distribution to create a BEAST2 input file
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = log_normal_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

---

create_tn93_site_model

Create a TN93 site model

Description

Create a TN93 site model

Usage

```
create_tn93_site_model(id = NA,
  gamma_site_model = create_gamma_site_model(),
  kappa_1_param = create_kappa_1_param(),
  kappa_2_param = create_kappa_2_param(),
  kappa_1_prior_distr = create_log_normal_distr(m = 1, s = 1.25),
  kappa_2_prior_distr = create_log_normal_distr(m = 1, s = 1.25),
  freq_equilibrium = "estimated")
```

Arguments

id	the IDs of the alignment (can be extracted from the FASTA filename using get_alignment_id)
gamma_site_model	a gamma site model, as created by create_gamma_site_model
kappa_1_param	the 'kappa 1' parameter, a numeric value. For advanced usage, use the structure as returned by create_kappa_1_param
kappa_2_param	the 'kappa 2' parameter, a numeric value. For advanced usage, use the structure as returned by create_kappa_2_param

kappa_1_prior_distr
: the distribution of the kappa 1 prior, which is a log-normal distribution (as created by create_log_normal_distr) by default

kappa_2_prior_distr
: the distribution of the kappa 2 prior, which is a log-normal distribution (as created by create_log_normal_distr) by default

freq_equilibrium
: the frequency in which the rates are at equilibrium are either estimated, empirical or all_equal. get_freq_equilibrium_names returns the possible values for freq_equilibrium

Value

a TN93 site_model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
tn93_site_model <- create_tn93_site_model(
  kappa_1_param = 2.0,
  kappa_2_param = 2.0
)

create_beast2_input_file(
  input_filename = get_fasta_filename(),
  "beast.xml",
  site_model = tn93_site_model
)
```

---

create_tree_prior *Internal function to create a tree prior*

Description

Internal function to create a tree prior

Usage

```
create_tree_prior(name, id, ...)
```

Arguments

name	the tree prior name. Can be any name in get_tree_prior_names
id	the ID of the alignment
...	specific tree prior parameters

Value

a tree_prior

Note

Prefer the use the named functions create_bd_tree_prior, create_cbs_tree_prior, create_ccp_tree_prior create_cep_tree_prior and create_yule_tree_prior instead

Author(s)

Richèl J.C. Bilderbeek

See Also

See create_bd_tree_prior, create_cbs_tree_prior, create_ccp_tree_prior create_cep_tree_prior and create_yule_tree_prior for more examples using those functions

Examples

```
beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_bd_tree_prior()
)
testit::assert(file.exists(beast2_input_file))

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_beautier_path("test_output_6.fas"),
  beast2_input_file,
  tree_prior = create_cbs_tree_prior()
)
testit::assert(file.exists(beast2_input_file))

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_ccp_tree_prior()
)
testit::assert(file.exists(beast2_input_file))

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_cep_tree_prior()
)
testit::assert(file.exists(beast2_input_file))

beast2_input_file <- tempfile(fileext = ".xml")
```

```
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior()
)
testit::assert(file.exists(beast2_input_file))
```

`create_tree_priors` *Creates all supported tree priors, which is a list of the types returned by* `create_bd_tree_prior`*,* `create_cbs_tree_prior`*,* `create_ccp_tree_prior`*,* `create_cep_tree_prior` *and* `create_yule_tree_prior`

Description

Creates all supported tree priors, which is a list of the types returned by `create_bd_tree_prior`, `create_cbs_tree_prior`, `create_ccp_tree_prior`, `create_cep_tree_prior` and `create_yule_tree_prior`

Usage

```
create_tree_priors()
```

Value

a list of tree_priors

Author(s)

Richèl J.C. Bilderbeek

Examples

```
tree_priors <- create_tree_priors()
testit::assert(beautier:::is_bd_tree_prior(tree_priors[[1]]))
testit::assert(beautier:::is_cbs_tree_prior(tree_priors[[2]]))
testit::assert(beautier:::is_ccp_tree_prior(tree_priors[[3]]))
testit::assert(beautier:::is_cep_tree_prior(tree_priors[[4]]))
testit::assert(beautier:::is_yule_tree_prior(tree_priors[[5]]))
```

`create_tree_priors_from_names` *Create tree priors from their names*

Description

Create tree priors from their names

Usage

```
create_tree_priors_from_names(tree_prior_names)
```

Arguments

`tree_prior_names`	one or more names of a tree prior, must be a name among those returned by `get_tree_prior_names`

Value

a tree prior, as can be created by using create_tree_prior

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_tree_prior to create a tree prior

Examples

```
names <- get_tree_prior_names()
tree_priors <- create_tree_priors_from_names(names)

for (i in seq_along(names)) {
  testthat::expect_equal(names[i], tree_priors[[i]]$name)
}
```

`create_tree_prior_from_name` *Create a tree prior from name*

Description

Create a tree prior from name

Usage

```
create_tree_prior_from_name(tree_prior_name)
```

Arguments

`tree_prior_name`	name of a tree prior, must be a name as returned by `get_tree_prior_names`

Value

a tree prior

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_tree_prior to create a tree prior

Examples

```
tree_prior_names <- get_tree_prior_names()
for (tree_prior_name in tree_prior_names) {
  tree_prior <- create_tree_prior_from_name(tree_prior_name)
  testthat::expect_equal(tree_prior_name, tree_prior$name)
}
```

`create_uniform_distr` *Create a uniform distribution*

Description

Create a uniform distribution

Usage

```
create_uniform_distr(id = NA, upper = Inf)
```

Arguments

`id`	the distribution's ID
`upper`	an upper limit of the uniform distribution. If the upper limits needs to be infinity, set `upper` to `Inf`.

Value

a uniform distribution

Author(s)

Richèl J.C. Bilderbeek

See Also

the function `create_distr` shows an overview of all supported distributions

Examples

```
uniform_distr <- create_uniform_distr()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = create_yule_tree_prior(
    birth_rate_distr = uniform_distr
  )
)
testit::assert(file.exists(beast2_input_file))
```

`create_yule_tree_prior` *Create a Yule tree prior*

Description

Create a Yule tree prior

Usage

```
create_yule_tree_prior(
  id = NA,
  birth_rate_distr = create_uniform_distr()
)
```

Arguments

`id`	the ID of the alignment
`birth_rate_distr`	the birth rate distribution, as created by a `create_distr` function

Value

a Yule tree_prior

Author(s)

Richèl J.C. Bilderbeek

See Also

An alignment ID can be extracted from its FASTA filename using `get_alignment_id`

Examples

```
yule_tree_prior <- create_yule_tree_prior()

beast2_input_file <- tempfile(fileext = ".xml")
create_beast2_input_file(
  input_filename = get_fasta_filename(),
  beast2_input_file,
  tree_prior = yule_tree_prior
)
testit::assert(file.exists(beast2_input_file))
```

default_parameters_doc *Documentation of parameters (for example,* create_param*. This function does nothing. It is intended to inherit documentation from.*

Description

Documentation of parameters (for example, create_param. This function does nothing. It is intended to inherit documentation from.

Usage

```
default_parameters_doc(estimate, id, lower, name, upper, value, ...)
```

Arguments

estimate	TRUE if this parameter is to be estimated by BEAST2, FALSE otherwise
id	the parameter's ID
lower	lowest possible value of the parameter. If the parameter is estimated, lower must be less than value
name	the parameters' name. Valid names can be found in get_param_names
upper	upper value of the parameter
value	value of the parameter
...	specific parameter parameters

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

default_params_doc *Documentation of general function arguments. This function does nothing. It is intended to inherit function argument documentation.*

Description

Documentation of general function arguments. This function does nothing. It is intended to inherit function argument documentation.

Usage

```
default_params_doc(alignment_id, bd_tree_prior, cbs_tree_prior,
  beauti_options, ccp_tree_prior, cep_tree_prior, chain_length,
  clock_model, clock_model_name, clock_model_names, clock_models,
  clock_prior_distr_id, crown_age, crown_ages, distr_id, fasta_filename,
  fasta_filenames, fixed_crown_age, fixed_crown_ages, gamma_site_model,
  group_sizes_dimension, gtr_site_model, has_non_strict_clock_model,
  has_tip_dating, hky_site_model, id, ids, inference_model,
  inference_models, initial_phylogenies, input_filename, input_filenames,
  is_monophyletic, jc69_site_model, mcmc, mrca_prior, mrca_priors,
  output_filename, param_id, phylogeny, posterior_crown_age,
  rln_clock_model, sequence_length, site_model, site_model_name,
  site_model_names, site_models, store_every, strict_clock_model,
  tipdates_filename, tn93_site_model, tree_prior, tree_prior_name,
  tree_prior_names, tree_priors, verbose, yule_tree_prior)
```

Arguments

alignment_id	ID of the alignment, as returned by get_alignment_id. Keep at NA to have it initialized automatically
bd_tree_prior	a Birth-Death tree prior, as created by create_bd_tree_prior
cbs_tree_prior	a Coalescent Bayesian Skyline tree prior, as returned by create_cbs_tree_prior
beauti_options	one BEAUti options object, as returned by create_beauti_options
ccp_tree_prior	a Coalescent Constant Population tree prior, as returned by create_ccp_tree_prior
cep_tree_prior	a Coalescent Exponential Population tree prior, as returned by create_cep_tree_prior
chain_length	length of the MCMC chain
clock_model	a clock model, as returned by create_clock_model
clock_model_name	name of a clock model, must be a name as returned by get_clock_model_names
clock_model_names	one or more names of a clock model, must be name among those returned by get_clock_model_names
clock_models	a list of one or more clock models, as returned by create_clock_model
clock_prior_distr_id	ID of an MRCA clock model's distribution. Keep at NA to have it initialized automatically
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
distr_id	a distributions' ID
fasta_filename	a FASTA filename. Use get_fasta_filename to obtain a testing FASTA filename.
fasta_filenames	One or more FASTA filenames. Use get_fasta_filename to obtain a testing FASTA filename.

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.

gamma_site_model
: a site model's gamma site model, as returned by create_gamma_site_model

group_sizes_dimension
: the group sizes' dimension, as used by the CBS tree prior (see create_cbs_tree_prior)

gtr_site_model
: a GTR site model, as returned by create_gtr_site_model

has_non_strict_clock_model
: boolean to indicate that the is already at least one non-strict (i.e. relaxed log-normal) clock model

has_tip_dating
: TRUE if the user has supplied tip dates, FALSE otherwise

hky_site_model
: an HKY site model, as returned by create_hky_site_model

id
: an alignment's IDs. An ID can be extracted from its FASTA filename with get_alignment_ids)

ids
: one or more alignments' IDs. IDs can be extracted from their FASTA filenames with get_alignment_ids)

inference_model
: an Bayesian phylogenetic inference model, as can be created by create_inference_model. An inference model is the complete model setup in which a site model, clock model, tree prior and more are specified.

inference_models
: a list of one or more inference models, as can be created by create_inference_model

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 phylo

input_filename
: A FASTA filename. Use get_fasta_filename to obtain a testing FASTA filename.

input_filenames
: One or more FASTA filenames. Use get_fasta_filename to obtain a testing FASTA filename.

is_monophyletic
: boolean to indicate monophyly is assumed in a Most Recent Common Ancestor prior, as returned by create_mrca_prior

jc69_site_model
: a JC69 site model, as returned by create_jc69_site_model

mcmc
: one MCMC as returned by create_mcmc

mrca_prior
: a Most Recent Common Ancestor prior, as returned by create_mrca_prior

mrca_priors a list of one or more Most Recent Common Ancestor priors, as returned by create_mrca_prior

output_filename
Name of the XML parameter file created by this function. BEAST2 uses this file as input.

param_id a parameter's ID

phylogeny a phylogeny of type phylo

posterior_crown_age
deprecated

rln_clock_model
a Relaxed Log-Normal clock model, as returned by create_rln_clock_model

sequence_length
a DNA sequence length, in base pairs

site_model a site model, as returned by create_site_model

site_model_name
name of a site model, must be a name as returned by get_site_model_names

site_model_names
one or more names of a site model, must be name among those returned by get_site_model_names

site_models one or more site models, as returned by create_site_model

store_every number of states the MCMC will process before the posterior's state will be saved to file. Use -1 or NA to use the default frequency.

strict_clock_model
a strict clock model, as returned by create_strict_clock_model

tipdates_filename
name of the file containing the tip dates. This file is assumed to have two columns, separated by a tab. The first column contains the taxa names, the second column contains the date.

tn93_site_model
a TN93 site model, as returned by create_tn93_site_model

tree_prior a tree priors, as returned by create_tree_prior

tree_prior_name
name of a tree prior, must be a name as returned by get_tree_prior_names

tree_prior_names
one or more names of a tree prior, must be a name among those returned by get_tree_prior_names

tree_priors one or more tree priors, as returned by create_tree_prior

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

yule_tree_prior
a Yule tree_prior, as created by create_yule_tree_prior

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

`distr_to_xml_beta` *Converts a beta distribution to XML*

Description

Converts a beta distribution to XML

Usage

```
distr_to_xml_beta(distr)
```

Arguments

`distr`	a beta distribution, as created by `create_beta_distr`)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

`distr_to_xml_exp` *Converts an exponential distribution to XML*

Description

Converts an exponential distribution to XML

Usage

```
distr_to_xml_exp(distr)
```

Arguments

`distr`	an exponential distribution, as created by `create_exp_distr`)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_gamma *Converts a gamma distribution to XML*

Description

Converts a gamma distribution to XML

Usage

```
distr_to_xml_gamma(distr)
```

Arguments

distr	a gamma distribution, as created by create_gamma_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_inv_gamma *Converts an inverse-gamma distribution to XML*

Description

Converts an inverse-gamma distribution to XML

Usage

```
distr_to_xml_inv_gamma(distr)
```

Arguments

distr	an inverse-gamma distribution, as created by create_inv_gamma_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_laplace *Converts a Laplace distribution to XML*

Description

Converts a Laplace distribution to XML

Usage

```
distr_to_xml_laplace(distr)
```

Arguments

distr	a Laplace distribution as created by create_laplace_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_log_normal *Converts a log-normal distribution to XML*

Description

Converts a log-normal distribution to XML

Usage

```
distr_to_xml_log_normal(distr)
```

Arguments

distr	a log-normal distribution, as created by create_log_normal_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_normal *Converts a normal distribution to XML*

Description

Converts a normal distribution to XML

Usage

```
distr_to_xml_normal(distr)
```

Arguments

distr	a normal distribution, as created by create_normal_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_one_div_x *Converts a 1/x distribution to XML*

Description

Converts a 1/x distribution to XML

Usage

```
distr_to_xml_one_div_x(distr)
```

Arguments

distr	a 1/x distribution, as created by create_one_div_x_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_poisson *Converts a Poisson distribution to XML*

Description

Converts a Poisson distribution to XML

Usage

```
distr_to_xml_poisson(distr)
```

Arguments

distr	a Poisson distribution, as created by create_poisson_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

distr_to_xml_uniform *Converts a uniform distribution to XML*

Description

Converts a uniform distribution to XML

Usage

```
distr_to_xml_uniform(distr)
```

Arguments

distr	a uniform distribution, as created by create_uniform_distr)

Value

the distribution as XML text

Author(s)

Richèl J.C. Bilderbeek

`fastas_to_phylos` *Create one or more random phylogenies.*

Description

Per FASTA file, one random phylogeny is created, with the same taxa names as that FASTA file. All phylogenies have the same crown age.

Usage

```
fastas_to_phylos(fasta_filenames, crown_age)
```

Arguments

`fasta_filenames`	One or more FASTA filenames. Use `get_fasta_filename` to obtain a testing FASTA filename.
`crown_age`	the crown age of the phylogeny

Value

a `multiPhylo` with as much phylogenies as there were FASTA filenames. Each phylogeny has the same taxa names as its corresponding FASTA file. All phylogenies have the same crown age.

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# Create two random phylogies, with
# - the same taxa names as the FASTA files
# - the desired crown age
fasta_filenames <- get_beautier_paths(
  c("anthus_aco.fas")
)
initial_phylogenies <- fastas_to_phylos(
  fasta_filenames,
  crown_age = 15
)
```

`fasta_to_phylo` *Create a random phylogeny, with the same taxa names as the FASTA file and the desired crown age*

Description

Create a random phylogeny, with the same taxa names as the FASTA file and the desired crown age

Usage

```
fasta_to_phylo(fasta_filename, crown_age)
```

Arguments

`fasta_filename`	a FASTA filename. Use `get_fasta_filename` to obtain a testing FASTA filename.
`crown_age`	the crown age of the phylogeny

Value

a a random phylogeny, with the same taxa names as the FASTA file and the desired crown age

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# Create a random phylogeny, with
# - the same taxa names as the FASTA file
# - the desired crown age
fasta_filename <- get_fasta_filename()
initial_phylogeny <- fasta_to_phylo(
  fasta_filename,
  crown_age = 15
)
```

`get_alignment_id` *Conclude the ID from a FASTA filename.*

Description

This is done in the same way as BEAST2 will do so.

Usage

```
get_alignment_id(fasta_filename, capitalize_first_char_id = FALSE)
```

Arguments

fasta_filename	a FASTA filename. Use get_fasta_filename to obtain a testing FASTA filename.
capitalize_first_char_id	if TRUE, the first character will be capitalized

Value

an alignment's ID

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# Path need not exist, use UNIX path as example
created <- get_alignment_id("/home/homer/anthus_aco_sub.fas")
expected <- "anthus_aco_sub"
testit::assert(created == expected)
```

---

get_alignment_ids *Get the alignment ID from one or more FASTA filenames.*

Description

This is done in the same way as BEAST2 does by default

Usage

```
get_alignment_ids(fasta_filenames)
```

Arguments

fasta_filenames	One or more FASTA filenames. Use get_fasta_filename to obtain a testing FASTA filename.

Value

the IDs from one or more FASTA files

Author(s)

Richèl J.C. Bilderbeek

Examples

```
created <- get_alignment_ids(
  get_beautier_paths(c("anthus_aco.fas", "anthus_nd2.fas"))
)
expected <- c(
  get_alignment_id(get_beautier_path("anthus_aco.fas")),
  get_alignment_id(get_beautier_path("anthus_nd2.fas"))
)
testit::assert(created == expected)
```

---

get_beautier_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_beautier_path(filename)
```

Arguments

filename	the file's name, without the path

Value

the full path of the filename

Author(s)

Richèl J.C. Bilderbeek

See Also

for more files, use get_beautier_paths

Examples

```
testit::assert(is.character(get_beautier_path("test_output_0.fas")))
testit::assert(is.character(get_beautier_path("anthus_aco.fas")))
testit::assert(is.character(get_beautier_path("anthus_nd2.fas")))
```

`get_beautier_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_beautier_paths(filenames)
```

Arguments

`filenames`	the files' names, without the path

Value

the filenames' full paths

Author(s)

Richèl J.C. Bilderbeek

See Also

Use get_beautier_path to get the path of one file

for one file, use `get_beautier_path`

Examples

```
testit::assert(
  length(
    get_beautier_paths(
      c("test_output_0.fas", "anthus_aco.fas", "anthus_nd2.fas")
    )
   ) == 3
 )
```

get_clock_model_names *Get the clock model names*

Description

Get the clock model names

Usage

```
get_clock_model_names()
```

Value

the clock model names

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_clock_models to create a list with all clock models

Examples

```
names <- beautier:::get_clock_model_names()
testit::assert("relaxed_log_normal" %in% names)
testit::assert("strict" %in% names)
```

get_crown_age *Obtain the crown age of a phylogeny.*

Description

The crown age of a phylogeny is the time between the present and the moment of at which the first diversification (resulting in two lineages) happened.

Usage

```
get_crown_age(phylogeny)
```

Arguments

phylogeny	The phylogeny to obtain the crown age of

Value

the crown age of the phylogeny

Author(s)

Richèl J.C. Bilderbeek

Examples

```
phylogeny <- ape::read.tree(text = "(a:15,b:15):1;")
created <- get_crown_age(phylogeny = phylogeny)
testit::assert(created == 15)
```

---

get_fasta_filename *Get the path of a FASTA file used in testing*

Description

Get the path of a FASTA file used in testing

Usage

```
get_fasta_filename()
```

Value

the path of a FASTA file used in testing

Author(s)

Richèl J.C. Bilderbeek

Examples

```
filename <- beautier::get_fasta_filename()
testit::assert(file.exists(filename))

create_beast2_input_file(
  input_filename = filename,
  "my_beast.xml"
)
```

`get_gamma_site_model_n_distrs` *Get the number of distributions in a gamma site model*

Description

Get the number of distributions in a gamma site model

Usage

```
get_gamma_site_model_n_distrs(gamma_site_model)
```

Arguments

`gamma_site_model` a site model's gamma site model, as returned by `create_gamma_site_model`

Value

the number of distributions a gamma site model has

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_gamma_site_model to create a gamma site model

Examples

```
gamma_site_model <- create_gamma_site_model()
n_distrs <- get_gamma_site_model_n_distrs(
  gamma_site_model
 )
testit::assert(n_distrs == 0)

gamma_site_model <- create_gamma_site_model(
  gamma_cat_count = 2,
  gamma_shape_prior_distr = create_exp_distr()
)
n_distrs <- get_gamma_site_model_n_distrs(gamma_site_model)
testit::assert(n_distrs == 1)
```

get_site_model_names *Get the site models' names*

Description

Get the site models' names

Usage

```
get_site_model_names()
```

Value

the site model names

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_site_models to get all site models

Examples

```
# Check all names
names <- get_site_model_names()
testit::assert("JC69" %in% names)
testit::assert("HKY" %in% names)
testit::assert("TN93" %in% names)
testit::assert("GTR" %in% names)
```

get_taxa_names *Extract the names of taxa from a file*

Description

Extract the names of taxa from a file

Usage

```
get_taxa_names(filename)
```

Arguments

filename	name of a FASTA file

Value

the taxa names

Author(s)

Richèl J.C. Bilderbeek

Examples

```
created <- get_taxa_names(get_beautier_path("anthus_aco_sub.fas"))
expected <- c(
  "61430_aco", "626029_aco", "630116_aco", "630210_aco", "B25702_aco"
 )
testit::assert(created == expected)
```

---

get_tree_prior_names *Get the tree prior names*

Description

Get the tree prior names

Usage

```
get_tree_prior_names()
```

Value

the tree prior names

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_tree_priors to get all tree priors

Examples

```
names <- beautier:::get_tree_prior_names()
testit::assert("birth_death" %in% names)
testit::assert("coalescent_bayesian_skyline" %in% names)
testit::assert("coalescent_constant_population" %in% names)
testit::assert("coalescent_exp_population" %in% names)
testit::assert("yule" %in% names)
```

has_mrca_prior *Determines if the inference model has an MRCA prior.*

Description

Will stop if the inference model is invalid

Usage

```
has_mrca_prior(inference_model)
```

Arguments

inference_model	an Bayesian phylogenetic inference model, as can be created by create_inference_model. An inference model is the complete model setup in which a site model, clock model, tree prior and more are specified.

Value

TRUE if the inference model has an MRCA prior, FALSE otherwise

Note

MRCA: 'Most Recent Common Ancestor'

Author(s)

Richèl J.C. Bilderbeek

See Also

- create_inference_model: create an inference model
- create_mrca_prior: create an MRCA prior

Examples

```
# No MRCA prior
inference_model <- create_inference_model(
  mrca_prior = NA
)
testthat::expect_false(has_mrca_prior(inference_model))

# A default MRCA prior
inference_model <- create_inference_model(
  mrca_prior = create_mrca_prior()
)
testthat::expect_true(has_mrca_prior(inference_model))
```

init_bd_tree_prior *Initializes a Birth-Death tree prior*

Description

Initializes a Birth-Death tree prior

Usage

```
init_bd_tree_prior(bd_tree_prior, distr_id, param_id)
```

Arguments

bd_tree_prior	a Birth-Death tree prior, as created by create_bd_tree_prior
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized Birth-Death tree prior

Author(s)

Richèl J.C. Bilderbeek

init_ccp_tree_prior *Initializes a Coalescent Constant Population tree prior*

Description

Initializes a Coalescent Constant Population tree prior

Usage

```
init_ccp_tree_prior(ccp_tree_prior, distr_id, param_id)
```

Arguments

ccp_tree_prior	a Coalescent Constant Population tree prior, as returned by create_ccp_tree_prior
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized Coalescent Constant Population tree prior

Author(s)

Richèl J.C. Bilderbeek

---

init_cep_tree_prior *Initializes a Coalescent Exponential Population tree prior*

Description

Initializes a Coalescent Exponential Population tree prior

Usage

```
init_cep_tree_prior(cep_tree_prior, distr_id, param_id)
```

Arguments

cep_tree_prior	a Coalescent Exponential Population tree prior, as returned by create_cep_tree_prior
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized Coalescent Exponential Population tree prior

Author(s)

Richèl J.C. Bilderbeek

---

init_gtr_site_model *Initializes a GTR site model*

Description

Initializes a GTR site model

Usage

```
init_gtr_site_model(gtr_site_model, distr_id = 0, param_id = 0)
```

Arguments

gtr_site_model	a GTR site model, as returned by create_gtr_site_model
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized GTR site model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
gtr_site_model <- create_gtr_site_model()
testit::assert(!beautier:::is_init_gtr_site_model(gtr_site_model))
gtr_site_model <- beautier:::init_gtr_site_model(gtr_site_model)
testit::assert(beautier:::is_init_gtr_site_model(gtr_site_model))
```

---

init_hky_site_model *Initializes an HKY site model*

Description

Initializes an HKY site model

Usage

```
init_hky_site_model(hky_site_model, distr_id = 0, param_id = 0)
```

Arguments

hky_site_model	an HKY site model, as returned by create_hky_site_model
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized HKY site model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
hky_site_model <- create_hky_site_model()
testit::assert(!beautier:::is_init_hky_site_model(hky_site_model))
hky_site_model <- beautier:::init_hky_site_model(hky_site_model)
testit::assert(beautier:::is_init_hky_site_model(hky_site_model))
```

init_jc69_site_model *Initializes a JC69 site model*

Description

Initializes a JC69 site model

Usage

```
init_jc69_site_model(jc69_site_model, distr_id = 0, param_id = 0)
```

Arguments

jc69_site_model	a JC69 site model, as returned by create_jc69_site_model
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized HKY site model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
hky_site_model <- create_hky_site_model()
testit::assert(!beautier:::is_init_hky_site_model(hky_site_model))
hky_site_model <- beautier:::init_hky_site_model(hky_site_model)
testit::assert(beautier:::is_init_hky_site_model(hky_site_model))
```

init_rln_clock_model *Initializes a Relaxed Log-Normal clock model*

Description

Initializes a Relaxed Log-Normal clock model

Usage

```
init_rln_clock_model(rln_clock_model, distr_id, param_id)
```

Arguments

rln_clock_model	a Relaxed Log-Normal clock model, as returned by create_rln_clock_model
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized Relaxed Log-Normal clock model

Author(s)

Richèl J.C. Bilderbeek

---

init_strict_clock_model

Initializes a strict clock model

Description

Initializes a strict clock model

Usage

```
init_strict_clock_model(strict_clock_model, distr_id, param_id)
```

Arguments

strict_clock_model	a strict clock model, as returned by create_strict_clock_model
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized strict clock model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
strict_clock_model <- create_strict_clock_model()
```

init_tn93_site_model *Initializes a TN93 site model*

Description

Initializes a TN93 site model

Usage

```
init_tn93_site_model(tn93_site_model, distr_id = 0, param_id = 0)
```

Arguments

tn93_site_model	a TN93 site model, as returned by create_tn93_site_model
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized TN93 site model

Author(s)

Richèl J.C. Bilderbeek

Examples

```
tn93_site_model <- create_tn93_site_model()
testit::assert(!beautier:::is_init_tn93_site_model(tn93_site_model))
tn93_site_model <- beautier:::init_tn93_site_model(tn93_site_model)
testit::assert(beautier:::is_init_tn93_site_model(tn93_site_model))
```

init_yule_tree_prior *Initializes a Yule tree prior*

Description

Initializes a Yule tree prior

Usage

```
init_yule_tree_prior(yule_tree_prior, distr_id, param_id)
```

Arguments

yule_tree_prior	a Yule tree_prior, as created by create_yule_tree_prior
distr_id	a distributions' ID
param_id	a parameter's ID

Value

an initialized Yule tree prior

Author(s)

Richèl J.C. Bilderbeek

---

is_bd_tree_prior *Determine if the object is a valid Birth Death tree prior*

Description

Determine if the object is a valid Birth Death tree prior

Usage

```
is_bd_tree_prior(x)
```

Arguments

x	an object, to be determined if it is a valid birth death tree prior

Value

TRUE if x is a valid birth death tree prior, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_bd_tree_prior to create a valid Birth-Death tree prior

Examples

```
testit::assert(is_bd_tree_prior(create_bd_tree_prior()))
testit::assert(!is_bd_tree_prior(create_cbs_tree_prior()))
testit::assert(!is_bd_tree_prior(create_ccp_tree_prior()))
testit::assert(!is_bd_tree_prior(create_cep_tree_prior()))
testit::assert(!is_bd_tree_prior(create_yule_tree_prior()))
```

`is_cbs_tree_prior` *Determine if the object is a valid constant coalescent Bayesian skyline prior*

Description

Determine if the object is a valid constant coalescent Bayesian skyline prior

Usage

```
is_cbs_tree_prior(x)
```

Arguments

x	an object, to be determined if it is a valid constant coalescent Bayesian skyline prior

Value

TRUE if x is a valid constant coalescent Bayesian skyline prior, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `create_cbs_tree_prior` to create a valid coalescent Bayes skyline tree prior

Examples

```
testit::assert(!is_cbs_tree_prior(create_bd_tree_prior()))
testit::assert( is_cbs_tree_prior(create_cbs_tree_prior()))
testit::assert(!is_cbs_tree_prior(create_ccp_tree_prior()))
testit::assert(!is_cbs_tree_prior(create_cep_tree_prior()))
testit::assert(!is_cbs_tree_prior(create_yule_tree_prior()))
```

`is_ccp_tree_prior` *Determine if the object is a valid constant coalescence population tree prior*

Description

Determine if the object is a valid constant coalescence population tree prior

Usage

```
is_ccp_tree_prior(x)
```

Arguments

`x`	an object, to be determined if it is a valid constant coalescence population tree prior

Value

TRUE if x is a valid constant coalescence population tree prior, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `create_ccp_tree_prior` to create a valid constant coalescence population tree prior

Examples

```
testit::assert(!is_ccp_tree_prior(create_bd_tree_prior()))
testit::assert(!is_ccp_tree_prior(create_cbs_tree_prior()))
testit::assert( is_ccp_tree_prior(create_ccp_tree_prior()))
testit::assert(!is_ccp_tree_prior(create_cep_tree_prior()))
testit::assert(!is_ccp_tree_prior(create_yule_tree_prior()))
```

`is_cep_tree_prior` *Determine if the object is a valid coalescent exponential population tree prior*

Description

Determine if the object is a valid coalescent exponential population tree prior

Usage

```
is_cep_tree_prior(x)
```

Arguments

`x`	an object, to be determined if it is a valid constant coalescent exponential population tree prior

Value

TRUE if x is a valid coalescent exponential population tree prior, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `create_cep_tree_prior` to create a valid coalescent exponential population tree prior

Examples

```
testit::assert(!is_cep_tree_prior(create_bd_tree_prior()))
testit::assert(!is_cep_tree_prior(create_cbs_tree_prior()))
testit::assert(!is_cep_tree_prior(create_ccp_tree_prior()))
testit::assert( is_cep_tree_prior(create_cep_tree_prior()))
testit::assert(!is_cep_tree_prior(create_yule_tree_prior()))
```

is_clock_model *Determine if the object is a valid clock_model*

Description

Determine if the object is a valid clock_model

Usage

```
is_clock_model(x)
```

Arguments

x	an object, to be determined if it is a clock_model

Value

TRUE if the clock_model is a valid clock_model, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

see create_clock_model for an overview of functions to create valid clock model

Examples

```
testit::assert(is_clock_model(create_strict_clock_model()))
testit::assert(is_clock_model(create_rln_clock_model()))
testit::assert(!is_clock_model("nonsense"))
```

is_init_beta_distr *Determine if x is an initialized beta distribution object as created by create_beta_distr*

Description

Determine if x is an initialized beta distribution object as created by create_beta_distr

Usage

```
is_init_beta_distr(x)
```

Arguments

x	the object to check if it is an initialized beta distribution object

Value

TRUE if x is an initialized beta distribution object

Author(s)

Richèl J.C. Bilderbeek

---

is_init_cep_tree_prior

Determine if x is an initialized Coalescent Exponential Population tree_prior object

---

Description

Determine if x is an initialized Coalescent Exponential Population tree_prior object

Usage

```
is_init_cep_tree_prior(x)
```

Arguments

x	the object to check if it is an initialized Coalescent Exponential Population tree prior object

Value

TRUE if x is an initialized Coalescent Exponential Population tree prior object

Author(s)

Richèl J.C. Bilderbeek

is_init_exp_distr *Determine if x is an initialized exponential distribution object as created by* create_exp_distr

Description

Determine if x is an initialized exponential distribution object as created by create_exp_distr

Usage

```
is_init_exp_distr(x)
```

Arguments

x	the object to check if it is an initialized exponential distribution object

Value

TRUE if x is an initialized exponential distribution object

Author(s)

Richèl J.C. Bilderbeek

is_init_gamma_distr *Determine if x is an initialized gamma distribution object*

Description

Determine if x is an initialized gamma distribution object

Usage

```
is_init_gamma_distr(x)
```

Arguments

x	the object to check if it is an initialized gamma distribution object

Value

TRUE if x is an initialized gamma distribution object

Author(s)

Richèl J.C. Bilderbeek

`is_init_inv_gamma_distr` *Determine if x is an initialized inverse-gamma distribution as created by* `create_inv_gamma_distr`

Description

Determine if x is an initialized inverse-gamma distribution as created by `create_inv_gamma_distr`

Usage

```
is_init_inv_gamma_distr(x)
```

Arguments

`x`	the object to check if it is an initialized inverse-gamma distribution

Value

TRUE if x is an initialized inverse-gamma distribution

Author(s)

Richèl J.C. Bilderbeek

`is_init_laplace_distr` *Determine if x is an initialized Laplace distribution as created by* `create_laplace_distr`

Description

Determine if x is an initialized Laplace distribution as created by `create_laplace_distr`

Usage

```
is_init_laplace_distr(x)
```

Arguments

`x`	the object to check if it is an initialized Laplace distribution

Value

TRUE if x is an initialized Laplace distribution

Author(s)

Richèl J.C. Bilderbeek

is_init_log_normal_distr

Determine if x is an initialized log_normal distribution object as created by create_log_normal_distr

Description

Determine if x is an initialized log_normal distribution object as created by create_log_normal_distr

Usage

```
is_init_log_normal_distr(x)
```

Arguments

x	the object to check if it is an initialized log_normal distribution object

Value

TRUE if x is an initialized log_normal distribution object

Author(s)

Richèl J.C. Bilderbeek

is_init_normal_distr

Determine if x is an initialized normal distribution object as created by create_normal_distr

Description

Determine if x is an initialized normal distribution object as created by create_normal_distr

Usage

```
is_init_normal_distr(x)
```

Arguments

x	the object to check if it is an initialized normal distribution object

Value

TRUE if x is an initialized normal distribution object

Author(s)

Richèl J.C. Bilderbeek

`is_init_one_div_x_distr` *Determine if x is an initialized one_div_x distribution object as created by* `create_one_div_x_distr`

Description

Determine if x is an initialized one_div_x distribution object as created by `create_one_div_x_distr`

Usage

```
is_init_one_div_x_distr(x)
```

Arguments

`x`	the object to check if it is an initialized one_div_x distribution object

Value

TRUE if x is an initialized one_div_x distribution object

Author(s)

Richèl J.C. Bilderbeek

`is_init_poisson_distr` *Determine if x is an initialized Poisson distribution object as created by* `create_poisson_distr`

Description

Determine if x is an initialized Poisson distribution object as created by `create_poisson_distr`

Usage

```
is_init_poisson_distr(x)
```

Arguments

`x`	the object to check if it is an initialized Poisson distribution object

Value

TRUE if x is an initialized Poisson distribution object

Author(s)

Richèl J.C. Bilderbeek

is_init_uniform_distr *Determine if x is an initialized uniform distribution object as created by* create_uniform_distr

Description

Determine if x is an initialized uniform distribution object as created by create_uniform_distr

Usage

```
is_init_uniform_distr(x)
```

Arguments

x	the object to check if it is an initialized uniform distribution object

Value

TRUE if x is an initialized uniform distribution object

Author(s)

Richèl J.C. Bilderbeek

is_mcmc *Determine if the object is a valid MCMC*

Description

Determine if the object is a valid MCMC

Usage

```
is_mcmc(x)
```

Arguments

x	an object, to be determined if it is a valid MCMC

Value

TRUE if x is a valid MCMC, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_mcmc to create an MCMC

Examples

```
testthat::expect_true(is_mcmc(create_mcmc()))
testthat::expect_true(is_mcmc(create_nested_sampling_mcmc()))
testthat::expect_false(is_mcmc("nonsense"))
```

---

is_mcmc_nested_sampling

Determine if the object is a valid Nested-Sampling MCMC, as used in [1]

Description

Determine if the object is a valid Nested-Sampling MCMC, as used in [1]

Usage

```
is_mcmc_nested_sampling(x)
```

Arguments

x	an object, to be determined if it is a valid MCMC

Value

TRUE if x is a valid Nested-Sampling MCMC, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

References

* [1] Patricio Maturana Russel, Brendon J Brewer, Steffen Klaere, Remco R Bouckaert; Model Selection and Parameter Inference in Phylogenetics Using Nested Sampling, Systematic Biology, 2018, syy050, https://doi.org/10.1093/sysbio/syy050

See Also

Use create_mcmc_nested_sampling

Examples

```
testthat::expect_false(is_nested_sampling_mcmc(create_mcmc()))
testthat::expect_true(
  is_nested_sampling_mcmc(create_nested_sampling_mcmc())
)
testthat::expect_false(is_nested_sampling_mcmc("nonsense"))
```

`is_one_na` *Determines if x is one NA*

Description

Determines if x is one NA

Usage

```
is_one_na(x)
```

Arguments

`x`	the object to be determined if it is one NA

Value

TRUE if x is one NA, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

Examples

```
testit::assert(is_one_na(NA))
testit::assert(!is_one_na(NULL))
testit::assert(!is_one_na(42))
testit::assert(!is_one_na("Hello"))
testit::assert(!is_one_na(3.14))
testit::assert(!is_one_na(c(NA, NA)))
```

is_phylo *Checks if the input is a phylogeny*

Description

Checks if the input is a phylogeny

Usage

```
is_phylo(x)
```

Arguments

x	input to be checked

Value

TRUE or FALSE

Author(s)

Richèl J.C. Bilderbeek

See Also

Use check_phylogeny to check for a phylogeny

Examples

```
phylogeny <- ape::read.tree(text = "(a:15,b:15):1;")
testit::assert(is_phylo(phylogeny))

testit::assert(!is_phylo("nonsense"))
testit::assert(!is_phylo(NA))
testit::assert(!is_phylo(NULL))
```

is_site_model *Determine if the object is a valid site_model*

Description

Determine if the object is a valid site_model

Usage

```
is_site_model(x)
```

Arguments

x	an object, to be determined if it is a site_model

Value

TRUE if the site_model is a valid site_model, FALSE otherwise

See Also

A site model can be created using `create_site_model`

Examples

```
# site models
testit::assert(is_site_model(create_gtr_site_model()))
testit::assert(is_site_model(create_hky_site_model()))
testit::assert(is_site_model(create_jc69_site_model()))
testit::assert(is_site_model(create_tn93_site_model()))

# other models
testit::assert(!is_site_model(create_strict_clock_model()))
testit::assert(!is_site_model(create_bd_tree_prior()))
testit::assert(!is_site_model(create_mcmc()))
```

---

is_tree_prior *Determine if an object is a valid tree prior*

Description

Determine if an object is a valid tree prior

Usage

```
is_tree_prior(x)
```

Arguments

x	an object

Value

TRUE if x is a valid tree_prior, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

tree priors can be created by create_tree_prior)

Examples

```
testit::assert(is_tree_prior(create_bd_tree_prior()))
testit::assert(is_tree_prior(create_yule_tree_prior()))
testit::assert(!is_tree_prior("nonsense"))
```

---

is_yule_tree_prior *Determine if the object is a valid Yule tree prior,*

Description

Determine if the object is a valid Yule tree prior,

Usage

```
is_yule_tree_prior(x)
```

Arguments

x	an object, to be determined if it is a valid Yule tree prior

Value

TRUE if x is a valid Yule tree prior, FALSE otherwise

Author(s)

Richèl J.C. Bilderbeek

See Also

Use create_yule_tree_prior to create a valid Yule tree prior

Examples

```
testit::assert(!is_yule_tree_prior(create_bd_tree_prior()))
testit::assert(!is_yule_tree_prior(create_cbs_tree_prior()))
testit::assert(!is_yule_tree_prior(create_ccp_tree_prior()))
testit::assert(!is_yule_tree_prior(create_cep_tree_prior()))
testit::assert( is_yule_tree_prior(create_yule_tree_prior()))
```

parameter_to_xml_alpha

Converts an alpha parameter to XML

Description

Converts an alpha parameter to XML

Usage

```
parameter_to_xml_alpha(parameter)
```

Arguments

parameter	an alpha parameter, a numeric value. For advanced usage, use the structure as created by create_alpha_param)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

parameter_to_xml_beta

Converts a beta parameter to XML

Description

Converts a beta parameter to XML

Usage

```
parameter_to_xml_beta(parameter)
```

Arguments

parameter	a beta parameter, a numeric value. For advanced usage, use the structure as created by create_beta_param)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_clock_rate`

Converts a `clockRate` *parameter to XML*

Description

Converts a `clockRate` parameter to XML

Usage

```
parameter_to_xml_clock_rate(parameter)
```

Arguments

`parameter`	a `clockRate` parameter, a numeric value. For advanced usage, use the structure as created by `create_clock_rate_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_kappa_1`

Converts a kappa 1 parameter to XML

Description

Converts a kappa 1 parameter to XML

Usage

```
parameter_to_xml_kappa_1(parameter)
```

Arguments

`parameter`	a kappa 1 parameter, a numeric value. For advanced usage, use the structure as created by `create_kappa_1_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_kappa_2`

Converts a kappa 2 parameter to XML

Description

Converts a kappa 2 parameter to XML

Usage

```
parameter_to_xml_kappa_2(parameter)
```

Arguments

`parameter`	a kappa 2 parameter, a numeric value. For advanced usage, use the structure as created by `create_kappa_2_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_lambda`

Converts a lambda parameter to XML

Description

Converts a lambda parameter to XML

Usage

```
parameter_to_xml_lambda(parameter)
```

Arguments

`parameter`	a lambda parameter, a numeric value. For advanced usage, use the structure as created by `create_lambda_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

---

parameter_to_xml_m *Converts a m parameter to XML*

Description

Converts a m parameter to XML

Usage

```
parameter_to_xml_m(parameter)
```

Arguments

parameter	a m parameter, a numeric value. For advanced usage, use the structure as created by create_m_param)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

---

parameter_to_xml_mean *Converts a mean parameter to XML*

Description

Converts a mean parameter to XML

Usage

```
parameter_to_xml_mean(parameter)
```

Arguments

parameter	a mean parameter, a numeric value. For advanced usage, use the structure as created by create_mean_param)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_mu` *Converts a mu parameter to XML*

Description

Converts a mu parameter to XML

Usage

```
parameter_to_xml_mu(parameter)
```

Arguments

`parameter`	a mu parameter, a numeric value. For advanced usage, use the structure as created by `create_mu_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_rate_ac`
Converts a 'rate AC' parameter to XML

Description

Converts a 'rate AC' parameter to XML

Usage

```
parameter_to_xml_rate_ac(parameter, which_name = "state_node")
```

Arguments

parameter	a 'rate AC' parameter, a numeric value. For advanced usage, use the structure as created by create_rate_ac_param)
which_name	the name, can be state_node or rate_name

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

---

parameter_to_xml_rate_ag

Converts a 'rate AG' parameter to XML

Description

Converts a 'rate AG' parameter to XML

Usage

```
parameter_to_xml_rate_ag(parameter, which_name = "state_node")
```

Arguments

parameter	a 'rate AG' parameter, a numeric value. For advanced usage, use the structure as created by create_rate_ag_param)
which_name	the name, can be state_node or rate_name

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

parameter_to_xml_rate_at *Converts a 'rate AT' parameter to XML*

Description

Converts a 'rate AT' parameter to XML

Usage

```
parameter_to_xml_rate_at(parameter, which_name = "state_node")
```

Arguments

parameter	a 'rate AT' parameter, a numeric value. For advanced usage, use the structure as created by create_rate_at_param)
which_name	the name, can be state_node or rate_name

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

parameter_to_xml_rate_cg *Converts a 'rate CG' parameter to XML*

Description

Converts a 'rate CG' parameter to XML

Usage

```
parameter_to_xml_rate_cg(parameter, which_name = "state_node")
```

Arguments

parameter	a 'rate CG' parameter, a numeric value. For advanced usage, use the structure as created by create_rate_cg_param)
which_name	the name, can be state_node or rate_name

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_rate_ct`

Converts a 'rate CT' parameter to XML

Description

Converts a 'rate CT' parameter to XML

Usage

```
parameter_to_xml_rate_ct(parameter, which_name = "state_node")
```

Arguments

`parameter`	a 'rate CT' parameter, a numeric value. For advanced usage, use the structure as created by `create_rate_ct_param`)
`which_name`	the name, can be `state_node` or `rate_name`

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_rate_gt`

Converts a 'rate GT' parameter to XML

Description

Converts a 'rate GT' parameter to XML

Usage

```
parameter_to_xml_rate_gt(parameter, which_name = "state_node")
```

Arguments

parameter	a 'rate GT' parameter, a numeric value. For advanced usage, use the structure as created by create_rate_gt_param)
which_name	the name, can be state_node or rate_name

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

---

parameter_to_xml_s *Converts a s parameter to XML*

Description

Converts a s parameter to XML

Usage

```
parameter_to_xml_s(parameter)
```

Arguments

parameter	a s parameter, a numeric value. For advanced usage, use the structure as created by create_s_param)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_scale`

Converts a scale parameter to XML

Description

Converts a scale parameter to XML

Usage

```
parameter_to_xml_scale(parameter)
```

Arguments

`parameter`	a scale parameter, a numeric value. For advanced usage, use the structure as created by `create_scale_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`parameter_to_xml_sigma`

Converts a sigma parameter to XML

Description

Converts a sigma parameter to XML

Usage

```
parameter_to_xml_sigma(parameter)
```

Arguments

`parameter`	a sigma parameter, a numeric value. For advanced usage, use the structure as created by `create_sigma_param`)

Value

the parameter as XML text

Author(s)

Richèl J.C. Bilderbeek

`yule_tree_prior_to_xml_prior_distr`
Creates the `prior` *section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Yule tree prior*

Description

Creates the `prior` section in the prior section of the prior section of the distribution section of a BEAST2 XML parameter file for a Yule tree prior

Usage

```
yule_tree_prior_to_xml_prior_distr(yule_tree_prior)
```

Arguments

`yule_tree_prior`
a Yule tree_prior, as created by `create_yule_tree_prior`

Author(s)

Richèl J.C. Bilderbeek

Examples

```
# <distribution id="posterior" spec="util.CompoundDistribution">
#     <distribution id="prior" spec="util.CompoundDistribution">
#       HERE, where the ID of the distribution is 'prior'
#     </distribution>
#     <distribution id="likelihood" ...>
#     </distribution>
# </distribution>
```

Index