Package ‘tracerer’

December 2, 2019

Type  Package
Title  Tracer from R
Version 2.0.2
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.
‘Tracer’ (<http://tree.bio.ed.ac.uk/software/tracer/>) is a GUI tool to parse and analyze the files generated by 'BEAST2'. This package provides a way to parse and analyze 'BEAST2' input files without active user input, but using R function calls instead.

License GPL-3
LazyData TRUE

Imports jsonlite, Rcpp, testit
Suggests ape, geiger, ggplot2, hunspell, knitr, phangorn, rbenchmark, reshape2, rmarkdown, spelling, stringr, testthat (>= 2.1.0)

VignetteBuilder knitr
RoxygenNote 6.1.1

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

BugReports https://github.com/ropensci/tracerer

LinkingTo Rcpp
Language en-US

Encoding UTF-8

NeedsCompilation yes

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),
R topics documented:

David Winter [rev] (David reviewed the package for rOpenSci, see https://github.com/ropensci/onboarding/issues/209)

Repository CRAN
Date/Publication 2019-12-02 12:10:02 UTC

R topics documented:

calc_act .......................................................... 3
calc_act_cpp .................................................... 4
calc_act_r ....................................................... 4
calc_ess .......................................................... 5
calc_esses ....................................................... 6
calc_hpd_interval ............................................. 7
calc_stderr_mean .............................................. 8
calc_std_error_of_mean_cpp ................................. 9
calc_summary_stats .......................................... 10
calc_summary_stats_trace ................................... 11
calc_summary_stats_traces .................................. 12
count_trees_in_file .......................................... 13
cs_std_dev ...................................................... 14
default_params_doc .......................................... 14
extract_operators_lines ..................................... 15
get_tracerer_path ............................................ 15
get_tracerer_paths ........................................... 16
is_posterior ................................................... 17
is_trees_file .................................................. 18
is_trees_posterior ........................................... 19
parse_beast_log ................................................ 19
parse_beast_output_files .................................... 20
parse_beast_posterior ........................................ 21
parse_beast_state_operators ................................ 22
parse_beast_trees ............................................ 23
remove_burn_in ............................................... 23
remove_burn_ins ............................................. 24
save_beast_estimates ........................................ 25
save_beast_trees ............................................. 25
tracerer .......................................................... 26

Index 27
Calculate the auto-correlation time, alternative implementation

Description

Calculate the auto-correlation time, alternative implementation

Usage

calc_act(trace, sample_interval)

Arguments

trace the values
sample_interval the interval in timesteps between samples

Value

the auto_correlation time

Author(s)

The original Java version of the algorithm was from Remco Bouckaert, ported to R and adapted by Richèl J.C. Bilderbeek

See Also

Java code can be found here: https://github.com/CompEvol/beast2/blob/9f040ed0357c4b946ea276a481a4c654ad4f76f/src/beast/core/util/ESS.java#L161 # nolint URLs can be long

Examples

trace <- sin(seq(from = 0.0, to = 2.0 * pi, length.out = 100))
act <- calc_act(
  trace = trace,
  sample_interval = 1
)
testthat::expect_equal(object = act, expected = 38.18202, tolerance = 0.01)
calc_act_cpp

*Calculate the auto correlation time from* [https://github.com/beast-dev/beast-mcmc/blob/800817772033c13061f026226e41128d21fd14f3/src/dr/inference/trace/TraceCorrelation.java#L159](https://github.com/beast-dev/beast-mcmc/blob/800817772033c13061f026226e41128d21fd14f3/src/dr/inference/trace/TraceCorrelation.java#L159) *# nolint*

**Description**

Calculate the auto correlation time from [https://github.com/beast-dev/beast-mcmc/blob/800817772033c13061f026226e41128d21fd14f3/src/dr/inference/trace/TraceCorrelation.java#L159](https://github.com/beast-dev/beast-mcmc/blob/800817772033c13061f026226e41128d21fd14f3/src/dr/inference/trace/TraceCorrelation.java#L159) *# nolint*

**Usage**

calc_act_cpp(sample, sample_interval)

**Arguments**

- **sample**
- **sample**
- **sample_interval**
- **sample interval**

**Value**

- the auto correlation time

**Author(s)**

Richèl J.C. Bilderbeek

---

calc_act_r

*Calculate the auto-correlation time using only R. Consider using calc_act instead, as it is orders of magnitude faster*

**Description**

Calculate the auto-correlation time using only R. Consider using `calc_act` instead, as it is orders of magnitude faster

**Usage**

calc_act_r(trace, sample_interval)
calc_ess

Arguments

- trace: the values
- sample_interval: the interval in timesteps between samples

Value

- the auto correlation time

Author(s)

The original Java version of the algorithm was from Remco Bouckaert, ported to R and adapted by Richèl J.C. Bilderbeek

See Also

Java code can be found here: https://github.com/CompEvol/beast2/blob/9f040ed0357c4b946ea276a481a4c654ad4fsrc/beast/core/util/ESS.java#L161 # nolint URLs can be long

Examples

```r
trace <- sin(seq(from = 0.0, to = 2.0 * pi, length.out = 100))
act <- calc_act_r(
  trace = trace,
  sample_interval = 1
)
testthat::expect_equal(object = act, expected = 38.18202, tolerance = 0.01)
```

calc_ess **Calculates the Effective Sample Size**

Description

Calculates the Effective Sample Size

Usage

calc_ess(trace, sample_interval)

Arguments

- trace: the values without burn-in
- sample_interval: the interval in timesteps between samples

Value

- the effective sample size
Author(s)

The original Java version of the algorithm was from Remco Bouckaert, ported to R and adapted by Richèl J.C. Bilderbeek

See Also

Java code can be found here: https://github.com/CompEvol/beast2/blob/9f040ed0357c4b946ea276a481a4c654ad4f/src/beast/core/util/ESS.java#L161 # nolint URLs can be long

Examples

```r
filename <- get_tracerer_path("beast2_example_output.log")

# Parse the file as-is and conclude the sampling interval
df <- parse_beast_log(filename)

# Only keep the parameter estimates,
# do not care about the sampling times anymore
estimates <- subset(df, select = -Sample) # nolint BEAST2 style

esses <- rep(NA, ncol(estimates))
burn_in_fraction <- 0.1
for (i in seq_along(estimates)) {
  # Trace with the burn-in still present
  trace_raw <- as.numeric(t(estimates[i]))
  # Trace with the burn-in removed
  trace <- remove_burn_in(
    trace = trace_raw,
    burn_in_fraction = 0.1
  )

  # Store the effective sample size
  esses[i] <- calc_ess(trace, sample_interval = sample_interval)
}

# Use the values that TRACER shows
expected_esses <- c(10, 10, 10, 10, 7, 10, 9, 6)
testit::assert(all(expected_esses - esses < 0.5))
```

calc_esses

Calculates the Effective Sample Sizes from a parsed BEAST2 log file

Description

Calculates the Effective Sample Sizes from a parsed BEAST2 log file
**calc_hpd_interval**

Calculate the Highest Probability Density of an MCMC trace that has its burn-in removed

**Description**

Calculate the Highest Probability Density of an MCMC trace that has its burn-in removed

**Usage**

```r
calc_hpd_interval(trace, proportion = 0.95)
```

**Usage**

```r
calc_esses(traces, sample_interval)
```

**Arguments**

- **traces**: a dataframe with traces with removed burn-in
- **sample_interval**: the interval in timesteps between samples

**Value**

the effective sample sizes

**Author(s)**

Richèl J.C. Bilderbeek

**Examples**

```r
# Parse an example log file
estimates_all <- parse_beast_log(
    get_tracerer_path("beast2_example_output.log")
)

# Remove burn-ins
estimates <- remove_burn_ins(estimates_all,
    burn_in_fraction = 0.1
)

# Calculate the effective sample sizes of all parameter estimates
esses <- calc_esses(
    estimates,
    sample_interval = 1000
)

expected <- c(10, 10, 10, 10, 7, 10, 9, 6)
testit::assert(all(esses == expected))
```
calc_stderr_mean

Arguments

trace a numeric vector of parameter estimates obtained from an MCMC run. Must have its burn-in removed
proportion the proportion of numbers within the interval. For example, use 0.95 for a 95 percentage interval

Value

a numeric vector, with at index 1 the lower boundary of the interval, and at index 2 the upper boundary of the interval

Author(s)

The original Java version of the algorithm was from J. Heled, ported to R and adapted by Richèl J.C. Bilderbeek

See Also

The function remove_burn_in removes a burn-in. The Java code that inspired this function can be found here: https://github.com/beast-dev/beast-mcmc/blob/98705c59db65e4f406a420bbade949aeecefe05d0/src/dr/stats/DiscreteStatistics.java#L317 # nolint URLs can be long

Examples

```r
estimates <- parse_beast_log(get_tracerer_path("beast2_example_output.log"))
  tree_height_trace <- remove_burn_in(
    estimates$TreeHeight,
    burn_in_fraction = 0.1
  )

  hpd_interval <- calc_hpd_interval(tree_height_trace, proportion = 0.95)
  testthat::expect_equivalent(0.453, hpd_interval[1], tolerance = 0.01)
  testthat::expect_equivalent(1.816, hpd_interval[2], tolerance = 0.01)
```
Value

the standard error of the mean

Author(s)

The original Java version of the algorithm was from Remco Bouckaert, ported to R and adapted by Richèl J.C. Bilderbeek

See Also

Java code can be found here: https://github.com/beast-dev/beast-mcmc/blob/800817772033c13061f026226e41128d21fd14f3/src/dr/inference/trace/TraceCorrelation.java#L159 # nolint URLs can be long

Examples

```r
trace <- sin(seq(from = 0.0, to = 2.0 * pi, length.out = 100))
stderr_mean <- calc_stderr_mean(trace)
testthat::expect_equal(stderr_mean, expected = 0.4347425, tolerance = 0.01)
```

---

calc_std_error_of_mean_cpp

*Calculates the standard error of the mean*

Description

Calculates the standard error of the mean

Usage

```r
calc_std_error_of_mean_cpp(sample)
```

Arguments

- `sample`: numeric vector of values

Value

the standard error of the mean

Author(s)

Richèl J.C. Bilderbeek
calc_summary_stats  Calculates the Effective Sample Sizes of one estimated variable’s trace.

Description
Calculates the Effective Sample Sizes of one estimated variable’s trace.

Usage

calc_summary_stats(traces, sample_interval)

Arguments
traces  one or more traces, supplies as either, (1) a numeric vector or, (2) a data frame of numeric values.
sample_interval  the interval (the number of state transitions between samples) of the MCMC run that produced the trace. Using a different sample_interval than the actually used sampling interval will result in bogus return values.

Value
the summary statistics of the traces. If one numeric vector is supplied, a list is returned with the elements listed below. If the traces are supplied as a data frame, a data frame is returned with the elements listed below as column names.
The elements are:

• mean: mean
• stderr_mean: standard error of the mean
• stdev: standard deviation
• variance: variance
• mode: mode
• geom_mean: geometric mean
• hpd_interval_low: lower bound of 95% highest posterior density
• hpd_interval_high: upper bound of 95% highest posterior density
• act: auto correlation time
• ess: effective sample size

Note
This function assumes the burn-in is removed. Use remove_burn_in (on a vector) or remove_burn_ins (on a data frame) to remove the burn-in.
calc_summary_stats_trace

Author(s)
Richèl J.C. Bilderbeek

See Also
Use `calc_summary_stats_trace` to calculate the summary statistics of one trace (stored as a numeric vector). Use `calc_summary_stats_traces` to calculate the summary statistics of more traces (stored as a data frame).

Examples

```r
estimates_all <- parse_beast_log(get_tracerer_path("beast2_example_output.log"))
estimates <- remove_burn_ins(estimates_all, burn_in_fraction = 0.1)

# From a single variable's trace
sum_stats_posterior <- calc_summary_stats(
estimates$posterior,
sample_interval = 1000
)
testit::assert("mean" %in% names(sum_stats_posterior))

# From all variables' traces
sum_stats <- calc_summary_stats(
estimates,
sample_interval = 1000
)
testit::assert("mean" %in% colnames(sum_stats))
```

calc_summary_stats_trace

*Calculates the Effective Sample Sizes of one estimated variable’s trace.*

Description

Calculates the Effective Sample Sizes of one estimated variable’s trace.

Usage

`calc_summary_stats_trace(trace, sample_interval)`

Arguments

- `trace`: a numeric vector of values. Assumes the burn-in is removed.
- `sample_interval`: the interval in timesteps between samples
calc_summary_stats_traces

Calculates the Effective Sample Sizes of the traces of multiple estimated variables.

Description

Calculates the Effective Sample Sizes of the traces of multiple estimated variables.

Usage

calc_summary_stats_traces(traces, sample_interval)

Arguments

traces a data frame with traces of estimated parameters. Assumes the burn-ins are removed.

sample_interval the interval in timesteps between samples

Value

the effective sample sizes

Examples

estimates_all <- parse_beast_log(get_tracerer_path("beast2_example_output.log"))
estimates <- remove_burn_ins(estimates_all, burn_in_fraction = 0.1)

sum_stats <- calc_summary_stats_trace(
estimates$posterior,
sample_interval = 1000
)

testit::assert("mean" %in% names(sum_stats))
count_trees_in_file

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

Use `remove_burn_ins` to remove the burn-ins of all traces

**Examples**

```r
estimates_all <- parse_beast_log(get_tracerer_path("beast2_example_output.log"))
estimates <- remove_burn_ins(estimates_all, burn_in_fraction = 0.1)

sum_stats <- calc_summary_stats_traces(
  estimates,
  sample_interval = 1000
)

testit::assert("mean" %in% colnames(sum_stats))
```

---

**count_trees_in_file**  
*Count the number of trees in a *.trees file*

**Description**

Count the number of trees in a *.trees file

**Usage**

```r
count_trees_in_file(trees_filename)
```

**Arguments**

- `trees_filename`  
  Name of a BEAST2 posterior *.trees file.

**Value**

- the number of trees

**Author(s)**

Richèl J.C. Bilderbeek

**See Also**

if the *.trees file is invalid, use `is_trees_file` with `verbose = TRUE` for the reason
### cs_std_dev

*Calculate the corrected sample standard deviation.*

**Description**

Calculate the corrected sample standard deviation.

**Usage**

```r
cs_std_dev(values)
```

**Arguments**

- **values**: numeric values

**Value**

The corrected sample standard deviation

**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**

```r
default_params_doc(trees_filename, verbose)
```

**Arguments**

- **trees_filename**: Name of a BEAST2 posterior .trees file.
- **verbose**: set to TRUE for more output

**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.
**extract_operators_lines**

*Author(s)*
Richèl J.C. Bilderbeek

---

**Description**
Extract the JSON lines out of a `.xml.state` with the unparsed BEAST2 MCMC operator acceptances file with the operators

**Usage**

```python
extract_operators_lines(filename)
```

**Arguments**

- `filename`: name of the BEAST2 `.xml.state` output file

**Value**

the JSON lines of a `.xml.state` file with the unparsed BEAST2 MCMC operator acceptances

**Author(s)**
Richèl J.C. Bilderbeek

---

**get_tracerer_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**

```python
get_tracerer_path(filename)
```

**Arguments**

- `filename`: the file’s name, without the path
get_tracerer_paths

Value
the full path to the filename

Author(s)
Richèl J.C. Bilderbeek

See Also
for more files, use get_tracerer_paths

Examples

```r
testit::assert(is.character(get_tracerer_path("beast2_example_output.log")))
testit::assert(is.character(get_tracerer_path("beast2_example_output.trees")))
testit::assert(is.character(get_tracerer_path("beast2_example_output.xml")))
testit::assert(is.character(get_tracerer_path("beast2_example_output.xml.state")))
```

---

**get_tracerer_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
```r
get_tracerer_paths(filenames)
```

Arguments

```r
filenames  the files' names, without the path
```

Value
the filenames’ full paths

Author(s)
Richèl J.C. Bilderbeek

See Also
for one file, use get_tracerer_path
is_posterior

Examples

testit::assert(
  length(
    get_tracerer_paths(
      c(
        "beast2_example_output.log",
        "beast2_example_output.trees",
        "beast2_example_output.xml",
        "beast2_example_output.xml.state"
      )
    )
  ) == 4
)

is_posterior Determines if the input is a BEAST2 posterior

Description

Determines if the input is a BEAST2 posterior

Usage

is_posterior(x)

Arguments

x the input

Value

TRUE if the input contains all information of a BEAST2 posterior. Returns FALSE otherwise.

Author(s)

Richèl J.C. Bilderbeek

Examples

trees_filename <- get_tracerer_path("beast2_example_output.trees")
log_filename <- get_tracerer_path("beast2_example_output.log")
posterior <- parse_beast_posterior(
  trees_filename = trees_filename,
  log_filename = log_filename
)

is_posterior(posterior)
is_trees_file

Measure if a file a valid BEAST2 .trees file

Description

Measure if a file a valid BEAST2 .trees file

Usage

is_trees_file(trees_filename, verbose = FALSE)

Arguments

trees_filename Name of a BEAST2 posterior .trees file.
verbose set to TRUE for more output

Value

TRUE if trees_filename is a valid .trees file

Author(s)

Richèl J.C. Bilderbeek

See Also

Most of the work is done by read.nexus

Examples

trees_filename <- get_tracerer_path("beast2_example_output.trees")
posterior <- parse_beast_trees(trees_filename)
testit::assert(is_trees_posterior(posterior))
library(testthat)

expect_true(is_trees_file(get_tracerer_path("beast2_example_output.trees")))
expect_true(is_trees_file(get_tracerer_path("unplottable_anthus_aco.trees")))
expect_true(is_trees_file(get_tracerer_path("anthus_2_4_a.trees")))
expect_true(is_trees_file(get_tracerer_path("anthus_2_4_b.trees")))
expect_false(is_trees_file(get_tracerer_path("mcbette_issue_8.trees"))))
is_trees_posterior

Determine if the input is a BEAST2 posterior, as parsed by parse_beast_trees

Description

Determines if the input is a BEAST2 posterior, as parsed by parse_beast_trees

Usage

is_trees_posterior(x)

Arguments

x the input

Value

TRUE or FALSE

Author(s)

Richèl J.C. Bilderbeek

parse_beast_log

 Parses a BEAST2 .log output file

Description

Parses a BEAST2 .log output file

Usage

parse_beast_log(filename)

Arguments

filename name of the BEAST2 .log output file

Value

 data frame with the parameter estimates

Author(s)

Richèl J.C. Bilderbeek
See Also

Use `remove_burn_ins` to remove the burn-in from the returned parameter estimates. Use `save_beast_estimates` to save the estimates to a .log file.

Examples

```r
log_filename <- get_tracerer_path("beast2_example_output.log")
estimates <- parse_beast_log(filename = log_filename)
expected_names <- c("Sample", "posterior", "likelihood", "prior", "treeLikelihood", "TreeHeight", "BirthDeath", "birthRate2", "relativeDeathRate2")
testit::assert(names(estimates) == expected_names)
```

parse_beast_output_files

`Parse all BEAST2 output files`

Description

Parse all BEAST2 output files

Usage

```r
parse_beast_output_files(log_filename, trees_filenames, state_filename)
```

Arguments

- `log_filename`: name of the BEAST2 .log file
- `trees_filenames`: name(s) of the BEAST2 .trees file(s) created. BEAST2 will create one .trees file per alignment
- `state_filename`: name of the BEAST2 .xml.state file created

Value

A list with the following elements:

- `estimates`: parameter estimates
- `[alignment_id]_trees`: the phylogenies in the BEAST2 posterior. `[alignment_id]` is the ID of the alignment.
- `operators`: the BEAST2 MCMC operator acceptances

Author(s)

Richèl J.C. Bilderbeek
parse_beast_posterior

See Also

Use `remove_burn_ins` to remove the burn-in from `out$estimates`

Examples

```r
trees_filenames <- get_tracerer_path("beast2_example_output.trees")
log_filename <- get_tracerer_path("beast2_example_output.log")
state_filename <- get_tracerer_path("beast2_example_output.xml.state")
out <- parse_beast_output_files(
  log_filename = log_filename,
  trees_filenames = trees_filenames,
  state_filename = state_filename
)
testit::assert("estimates" %in% names(out))
testit::assert("beast2_example_output_trees" %in% names(out))
testit::assert("operators" %in% names(out))
```

---

parse_beast_posterior  Parses BEAST2 output files to a posterior

Description

Parses BEAST2 output files to a posterior

Usage

`parse_beast_posterior(trees_filenames, log_filename)`

Arguments

- `trees_filenames`  one or more names of the BEAST2 .trees output files. A BEAST2 run will create as much .trees files as there are alignments
- `log_filename`   name of the BEAST2 .trees output file

Value

a list with the following elements:

- `estimates`: parameter estimates
- `item[alignment_id]_trees`: the phylogenies in the BEAST2 posterior. `[alignment_id]` is the ID of the alignment.

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `remove_burn_ins` to remove the burn-ins from the posterior’s estimates (posterior$estimates)
Examples

trees_filenames <- get_tracerer_path("beast2_example_output.trees")
log_filename <- get_tracerer_path("beast2_example_output.log")
posterior <- parse_beast_posterior(
  trees_filenames = trees_filenames,
  log_filename = log_filename
)
testit::assert(is_posterior(posterior))

parse_beast_state_operators

Parses a BEAST2 .xml.state output file to get only the operators acceptances

Description

Parses a BEAST2 .xml.state output file to get only the operators acceptances

Usage

parse_beast_state_operators(
  filename = get_tracerer_path("beast2_example_output.xml.state")
)

Arguments

filename name of the BEAST2 .xml.state output file

Value

data frame with all the operators’ success rates

Author(s)

Richèl J.C. Bilderbeek

Examples

xm_state_filename <- get_tracerer_path("beast2_example_output.xml.state")
estimates <- parse_beast_state_operators(filename = xm_state_filename)
expected_names <- c("operator", "p", "accept", "reject", "acceptFC",
                  "rejectFC", "rejectIv", "rejectOp")
expected_operator <- c("treeScaler.t", "treeRootScaler.t",
                      "UniformOperator.t", "SubtreeSlide.t", "narrow.t", "wide.t",
                      "WilsonBalding.t", "BirthRateScaler.t", "DeathRateScaler.t")
testit::assert(names(estimates) == expected_names)
#testit::assert(estimates$operator == expected_operators)
parse_beast_trees

Parses a BEAST2 .trees output file

Description

Parses a BEAST2 .trees output file

Usage

parse_beast_trees(filename)

Arguments

filename

name of the BEAST2 .trees output file

Value

the phylogenies in the posterior

Author(s)

Richèl J.C. Bilderbeek

See Also

Use save_beast_trees to save the phylogenies to a .trees file. Use is_trees_file with verbose = TRUE to find out why a file is invalid

Examples

```r
trees_filename <- get_tracerer_path("beast2_example_output.trees")
posterior <- parse_beast_trees(trees_filename)
testit::assert(is_trees_posterior(posterior))
```

remove_burn_in

Removed the burn-in from a trace

Description

Removed the burn-in from a trace

Usage

remove_burn_in(trace, burn_in_fraction)
Arguments

- `trace` the values
- `burn_in_fraction` the fraction that needs to be removed, must be \([0,1]\)

Value

the values with the burn-in removed

Author(s)

Richèl J.C. Bilderbeek

Examples

```r
# Create a trace from one to and including ten
v <- seq(1, 10)

# Remove the first ten percent of its values, 
# in this case removes the first value, which is one
w <- remove_burn_in(trace = v, burn_in_fraction = 0.1)

# Check that the result goes from two to ten
testit::assert(w == seq(2, 10))
```

Description

Removed the burn-ins from a data frame

Usage

```r
remove_burn_in(traces, burn_in_fraction = 0.1)
```

Arguments

- `traces` a data frame with traces
- `burn_in_fraction` the fraction that needs to be removed, must be \([0,1]\). Its default value of 10 as of Tracer

Value

the data frame with the burn-in removed

Author(s)

Richèl J.C. Bilderbeek
save_beast_estimates

Save the BEAST2 estimates as a BEAST2 .log file. There will be some differences: a BEAST2 .log file also saves the model as comments and formats the numbers in a way non-standard to R

Usage

save_beast_estimates(estimates, filename)

Arguments

estimates a data frame of BEAST2 parameter estimates
filename name of the .log file to save to

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use parse_beast_log to read a BEAST2 .log file

save_beast_trees

Save the BEAST2 trees as a BEAST2 .log file. There will be some differences: a BEAST2 .log file also saves the model as comments and formats the numbers in a way non-standard to R

Usage

save_beast_trees(trees, filename)
Arguments

- **trees**: BEAST2 posterior trees, of type `ape::multiPhylo`
- **filename**: name of the `.trees` file to save to

Value

nothing

Author(s)

Richèl J.C. Bilderbeek

See Also

Use `parse_beast_log` to read a BEAST2 `.log` file

**tracerer**

`tracerer`: A package to parse BEAST2 output files.

Description

`tracerer` allows to parse BEAST2 input files, using an R interface. 'tracerer' closely follows the functionality of Tracer, a GUI tool bundled with BEAST and BEAST2, including its default settings.

See Also

These are packages associated with `tracerer`:

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

If something is (still) missing from `tracerer`, the `coda` package may have the functionality you need.
Index

calc_act, 3, 4
calc_act_cpp, 4
calc_act_r, 4
calc_ess, 5
calc_esses, 6
calc_hpd_interval, 7
calc_std_error_of_mean_cpp, 9
calc_stderr_mean, 8
calc_summary_stats, 10
calc_summary_stats_trace, 11, 11
calc_summary_stats_traces, 11, 12
count_trees_in_file, 13
cs_std_dev, 14
default_params_doc, 14
extract_operators_lines, 15
get_tracerer_path, 15, 16
get_tracerer_paths, 16, 16
is_posterior, 17
is_trees_file, 13, 18, 23
is_trees_posterior, 19

parse_beast_log, 19, 25, 26
parse_beast_output_files, 20
parse_beast_posterior, 21
parse_beast_state_operators, 22
parse_beast_trees, 23

read.nexus, 18
remove_burn_in, 8, 10, 12, 23
remove_burn_ins, 10, 13, 20, 21, 24

save_beast_estimates, 20, 25
save_beast_trees, 23, 25

tracerer, 26
tracerer-package (tracerer), 26