

Package: rbi.helpers (via r-universe)

July 10, 2024

Version 0.4.0

Title 'rbi' Helper Functions

Author Sebastian Funk <sebastian.funk@lshtm.ac.uk>

Maintainer Sebastian Funk <sebastian.funk@lshtm.ac.uk>

Imports rbi (>= 0.10.0), data.table, lubridate, reshape2, Matrix

Suggests markdown, testthat, covr (>= 3.2.0), stringi, knitr,
rmarkdown

Description Contains a collection of helper functions to use with 'rbi', the R interface to 'LibBi', described in Murray et al. (2015) <[doi:10.18637/jss.v067.i10](https://doi.org/10.18637/jss.v067.i10)>. It contains functions to adapt the proposal distribution and number of particles in particle Markov-Chain Monte Carlo, as well as calculating the Deviance Information Criterion (DIC) and converting between times in 'LibBi' results and R time/dates.

License GPL-3

URL <https://libbi.org>, <https://github.com/sbfnk/rbi>,
<https://github.com/sbfnk/rbi.helpers>

BugReports <https://github.com/sbfnk/rbi.helpers/issues>

SystemRequirements libbi (>= 1.4.2)

LazyLoad no

RoxygenNote 7.2.3

NeedsCompilation no

Encoding UTF-8

VignetteBuilder knitr

Repository <https://epiforecasts.r-universe.dev>

RemoteUrl <https://github.com/sbfnk/rbi.helpers>

RemoteRef HEAD

RemoteSha 4df773a031651b82b84823e4a6064c37b4679708

Contents

acceptance_rate	2
adapt_particles	3
adapt_proposal	4
DIC	5
numeric_to_time	6
time_to_numeric	7
Index	8

acceptance_rate	<i>Compute acceptance rate</i>
-----------------	--------------------------------

Description

This function takes the provided `libbi` object which has been run, or a bi file, and returns a the acceptance rate

Usage

```
acceptance_rate(...)
```

Arguments

... parameters to `get_traces` (especially 'x')

Value

acceptance rate

Author(s)

Sebastian Funk

Examples

```
example_run <- rbi::bi_read(
  system.file(package = "rbi.helpers", "example_run.nc")
)
example_model_file <- system.file(package = "rbi", "PZ.bi")
example_bi <- rbi::attach_data(
  rbi::libbi(example_model_file), "output", example_run
)
acceptance_rate(example_bi)
```

adapt_particles	<i>Adapt the number of particles</i>
-----------------	--------------------------------------

Description

This function takes the provided `libbi` and runs MCMC at a single point (i.e., repeatedly proposing the same parameters), adapting the number of particles distribution until the variance of the log-likelihood crosses the value given as `target.variance` (1 by default).

Usage

```
adapt_particles(  
  x,  
  min = 1,  
  max = 1024,  
  target_variance = 1,  
  quiet = FALSE,  
  target.variance,  
  ...  
)
```

Arguments

<code>x</code>	a <code>libbi</code> object
<code>min</code>	minimum number of particles
<code>max</code>	maximum number of particles
<code>target_variance</code>	target log-likelihood variance; once this is crossed, the current number of particles will be used
<code>quiet</code>	if set to TRUE, will not provide running output of particle numbers tested
<code>target.variance</code>	deprecated; use <code>target_variance</code> instead
<code>...</code>	parameters for <code>libbi\$run</code>

Value

a `libbi` with the desired proposal distribution

Examples

```
example_obs <- rbi::bi_read(system.file(package="rbi", "example_dataset.nc"))  
example_model <- rbi::bi_model(system.file(package="rbi", "PZ.bi"))  
example_bi <- rbi::libbi(model = example_model, obs = example_obs)  
obs_states <- rbi::var_names(example_model, type = "obs")  
max_time <- max(vapply(example_obs[obs_states], function(x) {  
  max(x[["time"]])  
})
```

```

}, 0))
## Not run:
  adapted <- adapt_particles(example_bi, nsamples = 128, end_time = max_time)

## End(Not run)

```

adapt_proposal	<i>Adapt the proposal distribution of MCMC using the covariance of samples</i>
----------------	--------------------------------------------------------------------------------

Description

This function takes the provided `libbi` object and runs MCMC, adapting the proposal distribution until the desired acceptance rate is achieved. If a scale is given, it will be used to adapt the proposal at each iteration

Usage

```

adapt_proposal(
  x,
  min = 0,
  max = 1,
  scale = 2,
  max_iter = 10,
  adapt = c("size", "shape", "both"),
  size = FALSE,
  correlations = TRUE,
  truncate = TRUE,
  quiet = FALSE,
  ...
)

```

Arguments

<code>x</code>	link{libbi} object
<code>min</code>	minimum acceptance rate
<code>max</code>	maximum acceptance rate
<code>scale</code>	scale multiplier/divider for the proposal. If >1 this will be inverted.
<code>max_iter</code>	maximum of iterations (default: 10)
<code>adapt</code>	what to adapt; if "size" (default), the width of independent proposals will be adapted; if "shape", proposals will be dependent (following a multivariate normal) taking into account empirical correlations; if "both", the size will be adapted before the shape
<code>size</code>	(deprecated, use {adapt} instead) if TRUE (default: FALSE), the size of the (diagonal multivariate normal) proposal distribution will be adapted

correlations	(deprecated, use {adapt} instead) if TRUE (default: FALSE), the shape of the (diagonal multivariate normal) proposal distribution will be adapted according to the empirical covariance
truncate	if TRUE, the proposal distributions will be truncated according to the support of the prior distributions
quiet	if set to TRUE, will not provide running output of particle numbers tested
...	parameters for sample

Value

a [libbi](#) with the desired proposal distribution

Examples

```
example_obs <- rbi::bi_read(system.file(package="rbi", "example_dataset.nc"))
example_model <- rbi::bi_model(system.file(package="rbi", "PZ.bi"))
example_bi <- rbi::libbi(model = example_model, obs = example_obs)
obs_states <- rbi::var_names(example_model, type="obs")
max_time <- max(vapply(example_obs[obs_states], function(x) {
  max(x[["time"]])
}, 0))
# adapt to acceptance rate between 0.1 and 0.5
## Not run:
  adapted <- adapt_proposal(example_bi,
    nsamples = 100, end_time = max_time,
    min = 0.1, max = 0.5, nparticles = 256, correlations = TRUE
  )
## End(Not run)
```

 DIC

Compute Deviance Information Criterion (DIC) for a libbi model

Description

Computes the DIC of a libbi object containing Monte-Carlo samples. The effective number of parameters is calculated following Gelman et al., Bayesian Data Analysis: Second Edition, 2004, p. 182.

Usage

```
## S3 method for class 'libbi'
DIC(x, bootstrap = 0, ...)
```

Arguments

x	a libbi object
bootstrap	number of bootstrap samples to take, 0 to just take data
...	any parameters to be passed to 'bi_read' (e.g., 'burn')

Value

DIC

Author(s)

Sebastian Funk

Examples

```
example_run <- rbi::bi_read(  
  system.file(package = "rbi", "example_output.nc")  
)  
example_model_file <- system.file(package = "rbi", "PZ.bi")  
example_bi <- rbi::attach_data(  
  rbi::libbi(example_model_file), "output", example_run  
)  
DIC(example_bi)
```

`numeric_to_time`*Convert numeric times to actual times or dates*

Description

This function converts from numeric times (i.e., 0, 1, 2, ...) to actual times or dates

Usage

```
numeric_to_time(x, origin, unit, ...)
```

Arguments

<code>x</code>	a <code>libbi</code> object which has been run, or a list of data frames containing state trajectories (as returned by <code>bi_read</code>)
<code>origin</code>	the time origin, i.e. the date or time corresponding to time 0
<code>unit</code>	the unit of time that each time step corresponds to; this must be a unit understood by <code>lubridate::period</code> , optionally with a number in advance, e.g. "day" or "2 weeks" or "3 seconds"
<code>...</code>	any arguments for <code>bi_read</code> (e.g., file)

Value

a list of data frames as returned by `bi_read`, but with real times

time_to_numeric	<i>Convert actual times or dates to numeric times</i>
-----------------	-------------------------------------------------------

Description

This function converts from real times/dates to numeric times (0, 1, 2, ...)

Usage

```
time_to_numeric(x, origin, unit)
```

Arguments

x	a data frame containing a "time" column, or a list containing such data frames
origin	the time origin, i.e. the date or time corresponding to time 0
unit	the unit of time that each time step corresponds to; this must be a unit understood by <code>lubridate::period</code> , optionally with a number in advance, e.g. "day" or "2 weeks" or "3 seconds"

Value

a list of data frames that can be passed to `libbi`

Index

acceptance_rate, [2](#)
adapt_particles, [3](#)
adapt_proposal, [4](#)

DIC, [5](#)

get_traces, [2](#)

libbi, [2-6](#)

numeric_to_time, [6](#)

sample, [5](#)

time_to_numeric, [7](#)