Package ‘rbi’

April 17, 2020

Version 0.10.3
Title Interface to ‘LibBi’
Imports data.table, ncdf4, processx, reshape2
Suggests coda, covr (>= 3.2.0), R.rsp, stringi, testthat, ggplot2

Description Provides a complete interface to ‘LibBi’, a library for Bayesian inference (see <http://libbi.org> and <doi:10.18637/jss.v067.i10> for more information). This includes functions for manipulating ‘LibBi’ models, for reading and writing ‘LibBi’ input/output files, for converting ‘LibBi’ output to provide traces for use with the coda package, and for running ‘LibBi’ to conduct inference.

License GPL-3

URL https://github.com/sbfnk/RBi

BugReports https://github.com/sbfnk/RBi/issues
SystemRequirements LibBi (>= 1.4.2)
LazyLoad no
RoxygenNote 7.1.0
VignetteBuilder R.rsp
NeedsCompilation no

Author Pierre E. Jacob [aut],
Anthony Lee [ctb],
Lawrence M. Murray [ctb],
Sebastian Funk [aut, cre],
Sam Abbott [ctb]

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

Language en-GB
Repository CRAN
Date/Publication 2020-04-17 08:20:02 UTC
R topics documented:

rbi-package .................................................. 3
add_block .................................................... 4
attach_data ................................................... 4
bi_contents ................................................... 6
bi_dim_len ..................................................... 6
bi_file_summary .............................................. 7
bi_generate_dataset .......................................... 7
bi_model ........................................................ 8
bi_open .......................................................... 9
bi_read .......................................................... 9
bi_write ........................................................ 11
enable_outputs ............................................... 12
Equals.bi_model .............................................. 13
Extract.bi_model ............................................. 13
Extract_assign.bi_model ..................................... 14
extract_sample ............................................... 15
filter ............................................................. 15
fix ................................................................. 16
flatten ........................................................... 17
get_block ....................................................... 17
get_const ....................................................... 18
get_dims ......................................................... 18
get_name ....................................................... 19
get_traces ...................................................... 19
insert_lines .................................................... 20
join ............................................................... 21
libbi ............................................................. 21
logLik ............................................................ 22
optimise ......................................................... 23
option_list ..................................................... 23
option_string .................................................. 24
predict .......................................................... 24
print_log ........................................................ 25
read_libbi ....................................................... 25
remove_lines .................................................... 26
remove_vars ..................................................... 27
replace_all ..................................................... 27
rewrite .......................................................... 28
run ................................................................. 28
sample ........................................................... 31
sample_obs ..................................................... 31
save_libbi ....................................................... 32
set_name ........................................................ 33
simulate ........................................................ 33
summary ........................................................ 34
Unequals.bi_model ........................................... 35
rbi-package

Description

rbi is an interface to libbi, a library for Bayesian Inference

Details

The package includes a wrapper for the libbi script, allowing to launch the libbi command from within R. It also provides various utility functions to browse the output from libbi, for instance to plot the results.

The package is made of various components:

- A wrapper around libbi called libbi.
- A bi_model class that can be used to load and manipulate libbi models
- Functions to manipulate the results of the libbi command, which are stored in NetCDF files. Those functions allow to extract variables of interest.

Author(s)

Pierre E. Jacob <pierre.jacob.work@gmail.com>, Anthony Lee <awllee@gmail.com>, Lawrence Murray <lawrence.murray@csiro.au>, Sebastian Funk <sebastian.funk@lshtm.ac.uk>

References

http://libbi.org/

See Also

libbi

Examples

```r
example_output_file <- system.file(package="rbi", "example_output.nc")
bi_file_summary(example_output_file)
mu_sigma <- bi_read(example_output_file, c("mu", "sigma"))
bi_write("mu_sigma.nc", mu_sigma)

## examples for running libbi from rbi (will take a few minutes)
demo(PZ_generate_dataset)
demo(PZ_PMMH)
```
### add_block

#### Add a block to a LibBi model

**Description**

Add a block to a LibBi model. If that block exists, it will be removed first.

**Usage**

```r
## S3 method for class 'bi_model'
add_block(x, name, lines, options, ...)
```

**Arguments**

- `x`: a `bi_model` object
- `name`: name of the block
- `lines`: character vector, lines in the block
- `options`: any options to the block
- `...`: ignored

**Value**

a `bi_model` object containing the new block

---

### attach_data

#### Attach a new file or data set to a libbi object

**Description**

Adds an (output, obs, etc.) file to a `libbi` object. This is useful to recreate a `libbi` object from the model and output files of a previous run.

The `bi_write` options `append` and `overwrite` determine what exactly the file will contain at the end of this. If they are both `FALSE` (the default), any existing file will be ignored. If `append` is `TRUE`, the existing data in the file will be preserved, and any data set passed as `data` and not already in the file will be added. If `overwrite` is `TRUE`, existing data in the file will be preserved except for variables that exist in the passed data.

```r
demo(PZ_SMC2)
demo(PZ_filtering)
```
Usage

```r
## S3 method for class 'libbi'
attach_data(
  x,
  file,
  data,
  in_place = FALSE,
  append = FALSE,
  overwrite = FALSE,
  quiet = FALSE,
  time_dim = character(0),
  coord_dims = list(),
  ...
)
```

Arguments

- `x` - a `libbi` object
- `file` - the type of the file to attach, one of "output", "obs", "input" or "init"
- `data` - name of the file to attach, or a list of data frames that contain the outputs; it will be assumed that this is already thinned
- `in_place` - if `TRUE`, replace the file in place if it already exists in the `libbi` object; this can speed up the operation if `append=TRUE` as otherwise the file will have to be read and used again; it should be used with care, though, as it can render existing `libbi` objects invalid as the files they are pointing to are changed.
- `append` - if `TRUE`, will append variables if file exists; default: `FALSE`
- `overwrite` - if `TRUE`, will overwrite variables if file exists; default: `FALSE`
- `quiet` - if `TRUE`, will suppress the warning message normally given if `replace=TRUE` and the file exists already
- `time_dim` - the name of the time dimension, if one exists; default: "time"
- `coord_dims` - the names of the coordinate dimension, if any; should be a named list of character vectors, they are matched to variables names
- ... any options to `bi_write` (e.g., 'time_dim')

Value

- an updated `libbi` object

Examples

```r
bi <- libbi(model = system.file(package="rbi", "PZ.bi"))
example_output <- bi_read(system.file(package="rbi", "example_output.nc"))
bi <- attach_data(bi, "output", example_output)
```
**bi_contents**  

**Bi contents**

**Description**

This function gets the name of all the variables in the passed file, list or **libbi** object.

**Usage**

```r
bi_contents(read, ...)  
```

**Arguments**

- `read` either a path to a NetCDF file, or a NetCDF connection created using `nc_open`, or a **libbi** object from which to read the output
- `...` any parameters for **bi_open** (especially "file")

**Value**

character vector of variable names

**Examples**

```r
example_output_file <- system.file(package="rbi", "example_output.nc")
bi_contents(example_output_file)
```

---

**bi_dim_len**  

**NetCDF dimension length**

**Description**

This function returns the length of a dimension in a NetCDF file.

**Usage**

```r
bi_dim_len(filename, dim)  
```

**Arguments**

- `filename` path to a NetCDF file
- `dim` name of the dimension to check

**Value**

a number, the dimension length
NetCDF File Summary

Description

This function prints a little summary of the content of a NetCDF file, as well as its creation time. You can then retrieve variables of interest using \texttt{bi_read}.

Usage

\texttt{bi_file_summary(\ldots)}

Arguments

\ldots \hspace{1cm} \text{Any extra parameters to \texttt{bi_open}, especially \texttt{x} and \texttt{file}}

Value

No return value

Examples

\begin{verbatim}
example_output_file &lt; system.file(package="rbi", "example_output.nc")
bi_file_summary(example_output_file)
\end{verbatim}

Bi Generate Dataset

Description

This is a wrapper around \texttt{libbi sample --target joint --nsamples 1}, to generate a synthetic dataset from a model. Parameters can be passed via the 'init' option (see \texttt{run.libbi}, otherwise they are generated from the prior specified in the model. The end time should be specified using the "end_time" option. If this is not given, only a parameter set is sampled. Use the 'noutputs' or 'output_every' options to control the number of data points being generated. By default, output_every is set to 1.

Usage

\texttt{bi_generate_dataset(\ldots, output_every = 1)}

Arguments

\ldots \hspace{1cm} \text{arguments to be passed to \texttt{sample.libbi}, especially \texttt{model}, \texttt{end_time} and \texttt{seed}.}

\texttt{output_every} \hspace{1cm} \text{real; if given, noutputs will be set so that there is output every output_every time steps; if set to 0, only generate an output at the final time}
Value

a libbi object, the generated data set

Description

bi_model creates a model object for Rbi from a libbi file, URL or character vector. Once the instance is created, the model can be fed to a libbi object.

Usage

bi_model(filename, lines, ...)

Arguments

filename the file name of the model file
lines lines of the model (if no filename is given), a character vector
... ignored

Value

a {bi_model} object containing the newly created model

See Also

fix, insert_lines, remove_lines, replace_all, get_name, set_name, write_model

Examples

model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
**bi_open**  

**Bi open**

**Description**

This function opens an NetCDF file. The file can be specified as a string to the filepath, in which case a NetCDF connection is opened, or directly as a NetCDF connection.

**Usage**

```r
bi_open(x, file = "output")
```

**Arguments**

- `x`: either a path to a NetCDF file, or a NetCDF connection created using `nc_open`, or a `libbi` object from which to read the output
- `file`: file to open (out of "input", "init", "obs", "output"), if `x` is given as a `libbi` object; by default, will read output file

**Value**

an open NetCDF connection

---

**bi_read**  

**Bi Read**

**Description**

This function reads all variable from a NetCDF file or the output of a `libbi` object. The file can be specified as a string to the filepath, in which case a NetCDF connection is opened, or directly as a NetCDF connection.

**Usage**

```r
bi_read(
    x,
    vars,
    dims,
    model,
    type,
    file,
    missval_threshold,
    coord_dims = list(),
    vector,
    thin,
)```
Arguments

- **x**: either a path to a NetCDF file, or a NetCDF connection created using `nc_open`, or a `libbi` object from which to read the output
- **vars**: variables to read; if not given, all will be read
- **dims**: factors for dimensions
- **model**: model file or a `bi_model` object (if `x` is not a `libbi` object)
- **type**: vector of types of variable to read (out of "param", "state", "noise", "obs"). This needs `x` to be a `libbi` object or model to be specified
- **file**: which file to read (if `x` is given as a `libbi` object): one of "output" (default), "init", "input", "obs"
- **missval_threshold**: upper threshold for the likelihood
- **coord_dims**: any coord dimensions, given as a named list of character vectors, where each element corresponds to the variable of the same name, and the character vector are the coord dimensions
- **vector**: deprecated; if TRUE, will return results as vectors, not data.frames
- **thin**: thinning (keep only 1/thin of samples)
- **verbose**: if TRUE, will print variables as they are read
- **clear_cache**: if TRUE, will clear the cache and re-read the file even if cached data exists
- **init_to_param**: logical; if TRUE, convert states to initial values
- **burn**: number of initial samples to discard; default: 0
- **missval.threshold**: deprecated; use missval_threshold instead
- **init.to.param**: deprecated; use init_to_param instead

Value

a list of data frames and/or numbers that have been read

Examples

```r
example_output_file <- system.file(package="rbi", "example_output.nc")
d <- bi_read(example_output_file)
```
bi_write

Create (e.g., init or observation) files for LibBi

Description

This function creates (or appends to) a NetCDF file for LibBi from the given list of vectors and/or data frames. Since any files can be passed to libbi directly via the init, input and obs options, this is mostly used internally, this is mostly used internally.

Usage

bi_write(
  filename,
  variables,
  timed,
  append = FALSE,
  overwrite = FALSE,
  time_dim,
  coord_dims,
  dim_factors,
  value_column = "value",
  guess_time = FALSE,
  guess_coord = FALSE,
  verbose
)

Arguments

filename a path to a NetCDF file to write the variables into, which will be overwritten if it already exists. If necessary, ".nc" will be added to the file name
variables a list object, the names of which should be the variable names and values should be either single values or data frames
timed deprecated; timed variables should be given as data frames
append if TRUE, will append variables if file exists; default: FALSE
overwrite if TRUE, will overwrite variables if file exists; default: FALSE
time_dim the name of the time dimension, if one exists; default: "time"
coord_dims the names of the coordinate dimension, if any; should be a named list of character vectors, they are matched to variables names
dim_factors factors that dimensions have; this corresponds to the dims element of a libbi object
value_column if any variables are data frames, which column contains the values (default: "value")
guess_time whether to guess time dimension; this would be a numerical column in the data frame given which is not the value_column; only one such column must exist
enable_outputs

- **guess_coord**: whether to guess the coordinate dimension; this would be a column with varying value which is not the time or value column
- **verbose**: if TRUE, will print variables as they are read

**Details**

The list of variables must follow the following rules. Each element of the list must itself be one of:

1) a data frame with a `value_column` column (see option 'value_column') and any number of other columns indicating one or more dimensions
2) a numeric vector of length one, with no dimensions

The name of the list elements itself is used to create the corresponding variable in the NetCDF file.

**Value**

A list of the time and coord dims, and factors in extra dimensions, if any

**Examples**

```r
filename <- tempfile(pattern="dummy", fileext=".nc")
a <- 3
b <- data.frame(dim_a = rep(1:3, time = 2), dim_b = rep(1:2, each = 3), value = 1:6)
variables <- list(a=a, b=b)
bi_write(filename, variables)
bi_file_summary(filename)
```

---

**enable_outputs**

Enable outputting variables in a `bi_model`

**Description**

Any variable type given will have any `has_output=0` option removed in the given model.

**Usage**

`enable_outputs(x, type = "all")`

**Arguments**

- **x**: a `bi_model` object
- **type**: either "all" (default), or a vector of variable types that are to have outputs enabled

**Value**

the updated `bi_model` object

**See Also**

`bi_model`
Examples

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ[6] <- "param mu (has_output=0)"
PZ <- enable_outputs(PZ)
```

### Equals.bi_model

**Check if two models are equal**

**Description**

Ignores differences in the model name.

**Usage**

```r
## S3 method for class 'bi_model'
e1 == e2, ...
```

**Arguments**

- `e1`: a `bi_model`
- `e2`: a `bi_model`
- `...`: ignored

**Value**

TRUE or FALSE, depending on whether the models are equal or not

**Examples**

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ == PZ # TRUE
```

### Extract.bi_model

**Subset model lines**

**Description**

Extracts a subset of lines from the model.

**Usage**

```r
## S3 method for class 'bi_model'
x[i, ...]
```
### Extract_assign.bi_model

**Description**

Extracts a subset of lines from the model and assigns new character strings.

**Usage**

```r
## S3 replacement method for class 'bi_model'
x[i, ...] <- value
```

**Arguments**

- `x` A bi_model
- `i` A vector of line numbers
- `...` ignored
- `value` A vector of the same length as `i`, containing the replacement strings

**Value**

the updated bi_model object

**Examples**

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ[3:4] <- c("const e = 0.4", "const m_l = 0.05")
```
extract_sample

Extract a sample from a LibBi run.

Description
This function takes the provided libbi results and extracts a data frame.

Usage
extract_sample(x, np, ...)

Arguments
- x: a libbi object which has been run, or a list of data frames containing parameter traces (as returned by from bi_read)
- np: iteration to extract; if set to "last", the last sample will be extracted. If not given a random sample will be extracted
- ...: parameters to bi_read (e.g., dimensions)

Value
a list of data frames or numeric vectors containing parameters and trajectories

filter
Using the LibBi wrapper to filter

Description
The method filter launches libbi to filter state trajectories. See the options to run.libbi for how to specify the various components of sampling with LibBi, and the LibBi manual for all options that can be passed when the client is filter.

If x is given as a 'bi_model', a libbi object will be created from the model For the help page of the base R filter function, see filter.

Usage
## S3 method for class 'libbi'
filter(x, ...)

## S3 method for class 'bi_model'
filter(x, ...)

Arguments
- x: a libbi or bi_model object, or the name of a file containing the model
- ...: options to be passed to run.libbi
Value

an updated libbi object

fix

Fix noise term, state or parameter of a libbi model

Description

Replaces all variables with fixed values as given; note that this will not replace differential equations and lead to an error if applied to states that are changed inside an "ode" block.

For the help page of the base R fix function, see fix.

Usage

## S3 method for class 'bi_model'
fix(x, ...)

Arguments

x a bi_model object

... values to be assigned to the (named) variables

Value

the updated bi_model object

See Also

bi_model

Examples

model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ <- fix(PZ, alpha = 0)
### flatten

**Flatten list of data frames**

This function takes a list of data frames (such as, for example, returned by `bi_read`) and converts it to a flat data frame.

#### Description

Flatten list of data frames This function takes a list of data frames (such as, for example, returned by `bi_read`) and converts it to a flat data frame.

#### Usage

```r
flatten(x)
```

#### Arguments

- **x**
  
  The list of data frames

#### Value

A data frame containing the flattened data.

---

### get_block

*Get the contents of a block in a LibBi model*

#### Description

Returns the contents of a block in a LibBi model as a character vector of lines.

#### Usage

```r
## S3 method for class 'bi_model'
get_block(x, name, shell = FALSE, ...)
```

#### Arguments

- **x**
  
  A `bi_model` object

- **name**
  
  Name of the block

- **shell**
  
  If TRUE (default: FALSE), will return the shell (i.e., the definition of the block) as well as content; this is useful, e.g., to see options passed to a transition or ode block

- **...**
  
  Ignored

#### Value

A character vector of the lines in the block.
get_const

Get constants in a LibBi model

Description
Get constants contained in a LibBi model and their values. This will attempt to evaluate any calculation on the right hand side. Failing that, it will be returned verbatim.

Usage
get_const(model)

Arguments

model: a bi_model object

Value
a list of constants (as names) and their values

get_dims

Get dimensions in a LibBi model

Description
Get dimensions contained in a LibBi model and their sizes

Usage
get_dims(model, type)

Arguments

model: a bi_model object
type: a character vector of one or more types

Value
a list of dimensions (as names) and their sizes
get_name

Get the name of a bi model

Description

Extracts the name of a bi model (first line of the .bi file).

Usage

```r
## S3 method for class 'bi_model'
get_name(x, ...)
```

Arguments

- `x`: a `bi_model` object
- `...`: ignored

Value

a character string, the name of the model

See Also

`bi_model`

Examples

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
get_name(PZ)
```

get_traces

Get the parameter traces

Description

This function takes the provided `libbi` object which has been run and returns a data frame with the parameter traces.

Usage

```r
get_traces(x, model, burnin, all = FALSE, ...)
```
Arguments

x
   a `libbi` object which has been run, or a list of data frames containing parameter traces (as returned by `bi_read`); if it is not a `libbi` object, either 'all' must be TRUE or a model given

model
   a model to get the parameter names from; not needed if 'run' is given as a `libbi` object or 'all' is set to TRUE

burnin
   proportion of iterations to discard as burn-in (if between 0 and 1), or number of samples to discard (if >1)

all
   whether all variables in the run file should be considered (otherwise, just parameters)

... parameters to `bi_read` (e.g., dimensions)

Value

a ata frame with parameter traces; this can be fed to coda routines

---

**insert_lines**

*Insert lines in a LibBi model*

Description

Inserts one or more lines into a libbi model. If one of before or after is given, the line(s) will be inserted before or after a given line number or block name, respectively. If one of at_beginning of or at_end_of is given, the lines will be inserted at the beginning/end of the block, respectively.

Usage

```r
## S3 method for class 'bi_model'
insert_lines(x, lines, before, after, at_beginning_of, at_end_of, ...)
```

Arguments

x
   a `bi_model` object

lines
   vector or line(s)

before
   line number before which to insert line(s)

after
   line number after which to insert line(s)

at_beginning_of
   block at the beginning of which to insert lines(s)

at_end_of
   block at the end of which to insert lines(s)

... ignored

Value

the updated `bi_model` object
See Also
   bi_model

Examples
   model_file_name <- system.file(package="rbi", "PZ.bi")
   PZ <- bi_model(filename = model_file_name)
   PZ <- insert_lines(PZ, lines = "noise beta", after = 8)

Description
   This function can be used to join multiple libbi objects into one (e.g., parallel MCMC runs into
   one long change)

Usage
   ## S3 method for class 'libbi'
   join(x, ...)

Arguments
   x       a libbi object
   ...     ignored

Value
   an joined libbi object

libbi       LibBi Wrapper

Description
   libbi allows to call LibBi. Upon creating a new libbi object, the following arguments can be
given. Once the instance is created, LibBi can be run through the sample, filter, or optimise, or
rewrite methods. Note that libbi objects can be plotted using plot if the rbi.helpers package
is loaded.

Usage
   libbi(model, path_to_libbi, dims, use_cache = TRUE, ...)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>model</code></td>
<td>either a character vector giving the path to a model file (typically ending in &quot;.bi&quot;), or a <code>bi_model</code> object</td>
</tr>
<tr>
<td><code>path_to_libbi</code></td>
<td>path to LibBi binary; by default it tries to locate the <code>libbi</code> binary using the <code>which</code> Unix command, after having loaded &quot;<del>/.bashrc&quot; if present; if unsuccessful it tries &quot;</del>/PathToBiBin/libbi&quot;; if unsuccessful again it fails.</td>
</tr>
<tr>
<td><code>dims</code></td>
<td>any named dimensions, as list of character vectors</td>
</tr>
<tr>
<td><code>use_cache</code></td>
<td>logical; whether to use the cache (default: true)</td>
</tr>
<tr>
<td>...</td>
<td>options passed to <code>run.libbi</code></td>
</tr>
</tbody>
</table>

Value

a new `libbi` object

See Also

`sample`, `filter`, `optimise`, `rewrite`

Examples

```r
bi_object <- libbi(model = system.file(package="rbi", "PZ.bi"))
```

Description

The method `logLik` extracts the log-likelihood of a `libbi` object. This can be done, for example, after a call to `sample` to inspect the chain log-likelihoods.

For the help page of the base R `logLik` function, see `logLik`.

Usage

```r
## S3 method for class 'libbi'
logLik(object, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object</code></td>
<td>a <code>libbi</code> object</td>
</tr>
<tr>
<td><code>...</code></td>
<td>options to be passed to <code>run.libbi</code></td>
</tr>
</tbody>
</table>

Value

a vector of log-likelihood
Using the LibBi wrapper to optimise

Description

The method optimise launches libbi to optimise the parameters with respect to the likelihood or posterior distribution. See the options to run.libbi for how to specify the various components of sampling with LibBi, and the LibBi manual for all options that can be passed when the client is optimise.

If x is given as a 'bi_model', a libbi object will be created from the model For the help page of the base R optimise function, see optimise.

Usage

```r
## S3 method for class 'libbi'
optimise(x, ...)

## S3 method for class 'bi_model'
optimise(x, ...)
```

Arguments

- `x` a libbi or link{bi_model} object, or the name of a file containing the model
- `...` options to be passed to run.libbi

Value

an updated libbi object

option_list

Convert string to option list

Description

This function is used to convert an option string into a list of options. If a list is given, it will be kept as is

Usage

option_list(...)

Arguments

- `...` any number of strings to convert
### Value

A list of options and values

---

### option_string  
*Convert Options*

#### Description

This function is used to convert a list of options into an options string. If a string is given, it will be taken as such.

#### Usage

```r
option_string(...)  
```

#### Arguments

...  
Any number of lists of options, or strings (which will be left unmodified). If lists are given, later arguments will override earlier ones.

---

### predict  
*Using the LibBi wrapper to predict*

#### Description

The method `predict` is an alias for `sample(target="prediction")`. Usually, an `init` object or file should be given containing posterior samples.

For the help page of the base R `optimise` function, see `optimise`.

#### Usage

```r
## S3 method for class 'libbi'
predict(x, ...)  
```

#### Arguments

- `x`  
  A `libbi` object

- ...  
  Any arguments to be passed to `sample`

#### Value

An updated `libbi` object
**print_log**

*Print the log file a libbi object*

**Description**

This is useful for diagnosis after a libbi run.

**Usage**

```
print_log(x)
```

**Arguments**

- `x` a libbi object, or the name of the log file of a libbi run.

**Value**

nothing (invisible NULL)

---

**read_libbi**

*Read results of a LibBi run from an RDS file or from a folder. This completely reconstructs the saved LibBi object*

**Description**

This reads all options, files and outputs of a LibBi run from a specified RDS file or folder (if `split` = TRUE has been used with `save_libbi`).

**Usage**

```
read_libbi(name, ...)
```

**Arguments**

- `name` name of the RDS file(s) to read
- `...` any extra options to pass to libbi when creating the new object

**Value**

a new libbi object
remove_lines

Remove line(s) and/or block(s) in a libbi model

Description

Removes one or more lines in a libbi model.

Usage

```r
## S3 method for class 'bi_model'
remove_lines(
  x,            
  what,         
  only,         
  type = c("all", "assignment", "sample"),
  preserve_shell = FALSE,
  ...            
)
```

Arguments

- `x` a `bi_model` object
- `what` either a vector of line number(s) to remove, or a vector of blocks to remove (e.g., "parameter")
- `only` only remove lines assigning given names (as a vector of character strings)
- `type` which types of lines to remove, either "all", "sample" (i.e., lines with a "~") or "assignment" (lines with a "<-" or ":=") (default: "all")
- `preserve_shell` if TRUE (default: FALSE), preserve the definition of a block even if all lines are removed; this is useful to preserve options passed to a transition or ode block
- `...` ignored

Value

the updated `bi_model` object

See Also

`bi_model`

Examples

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ <- remove_lines(PZ, 2)
```
**remove_vars**  
*Remove variables*

**Description**
Removes variables from the left-hand side of a model
Used by `fix` and `to_input`

**Usage**
```r
remove_vars(x, vars)
```

**Arguments**
- `x` a `bi_model` object
- `vars` vector of variables to remove

**Value**
a bi model object of the new model
the updated `bi_model` object

**See Also**
`bi_model`

---

**replace_all**  
*Replace all instances of a string with another in a model*

**Description**
Takes every occurrence of one string and replaces it with another

**Usage**
```r
## S3 method for class 'bi_model'
replace_all(x, from, to, ...)
```

**Arguments**
- `x` a `bi_model` object
- `from` string to be replaced (a regular expression)
- `to` new string (which can refer to the regular expression given as `from`)
- `...` ignored
Value

the updated bi_model object

See Also

bi_model

---

**rewrite**

*Using the LibBi wrapper to rewrite*

**Description**

The method `rewrite` launches LibBi to rewrite a model to inspect its internal representation in LibBi.

If `x` is given as a 'bi_model', a `libbi` object will be created from the model.

**Usage**

```r
## S3 method for class 'libbi'
rewrite(x, ...)
```

```r
## S3 method for class 'bi_model'
rewrite(x, ...)
```

**Arguments**

- `x` a `libbi` or `bi_model` object, or the name of a file containing the model
- `...` options to be passed to `run.libbi`

**Value**

a re-written `bi_model` object

---

**run**

*Using the LibBi wrapper to launch LibBi*

**Description**

The method `run` launches LibBi with a particular set of command line `#` arguments. Normally, this function would not be run by the user, but instead one of the client functions `sample`, `filter`, or `optimise`, or `rewrite`, which pass any options on to `run`. Note that any options specified here are stored in the `libbi` object and do not have to be specified again if another command is run on the object.
## S3 method for class 'libbi'
run(
  x,
  client,
  proposal = c("model", "prior"),
  model,
  fix,
  options,
  config,
  log_file_name = character(0),
  init,
  input,
  obs,
  time_dim = character(0),
  coord_dims = list(),
  working_folder,
  output_all = FALSE,
  sample_obs = FALSE,
  thin,
  output_every,
  chain = TRUE,
  seed = TRUE,
  debug = FALSE,
  ...
)

### Arguments

- **x**: a `libbi` object; if this is not given, an empty `libbi` object will be created
- **client**: client to pass to LibBi
- **proposal**: proposal distribution to use; either "model" (default: proposal distribution in the model) or "prior" (propose from the prior distribution)
- **model**: either a character vector giving the path to a model file (typically ending in ".bi"), or a bi_model object; by default, will use any model given in x
- **fix**: any variable to fix, as a named vector
- **options**: deprecated; pass options directly, see documentation for ...
- **config**: path to a configuration file, containing multiple arguments
- **log_file_name**: path to a file to text file to report the output of LibBi; if set to an empty vector (character(0)) or an empty string (""), which is the default, a temporary log file will be generated
- **init**: initialisation of the model, either supplied as a list of values and/or data frames, or a (netcdf) file name, or a `libbi` object which has been run (in which case the output of that run is used). If the object given as x has been run before, it will be used here with init=npc set to the last iteration of the previous run, unless init is given explicitly.
input of the model, either supplied as a list of values and/or data frames, or a
(netcdf) file name, or a `libbi` object which has been run (in which case the
output of that run is used as input)

observations of the model, either supplied as a list of values and/or data frames,
or a (netcdf) file name, or a `libbi` object which has been run (in which case the
output of that run is used as observations)

The time dimension in any R objects that have been passed (init, input) and
obs; if NULL (default), will be guessed from the given observation

The coord dimension(s) in any obs R objects that have been passed; if NULL
(default), will be guessed from the given observation file given

deprecated; path to a folder from which to run LibBi; default to a temporary
folder. Use the build_dir option instead, which will be translated into the
--build-dir option of LibBi instead

deprecated; if set to TRUE, all parameters, states and observations will be saved;
good for debugging

deprecated; if set to TRUE, will sample observations. Use the sample_obs
function instead.

any thinning of MCMC chains (1 means all will be kept, 2 skips every other
sample etc.); note that LibBi itself will write all data to the disk. Only when the
results are read in with `bi_read` will thinning be applied.

real; if given, noutputs will be set so that there is output every output_every
time steps; if set to 0, only generate an output at the final time

logical; if set to TRUE and x has been run before, the previous output file will
be used as init file, and init-np will be set to the last iteration of the previous
run (unless target=="prediction"). This is useful for running inference chains.

Either a number (the seed to supply to LibBi), or a logical variable: TRUE if a
seed is to be generated for RBi, FALSE if LibBi is to generate its own seed

logical; if TRUE, print more verbose messages and write all variables to the
output file, irrespective of their setting of 'has_output'

list of additional arguments to pass to the call to LibBi. Any arguments starting
with 'enable'/"disable" can be specified as boolean (e.g., 'assert=TRUE' or
'cuda=TRUE'). Any 'dry-' options can be specified with a "dry" argument,
e.g., 'dry="parse"'. Any options that would be specified with 'with'/"without"
can be specified as character vector to an option named 'with'/'without', respec-
tively, e.g. with="transform-obs-to-state".

an updated `libbi` object, except if client is 'rewrite', in which case invisible NULL will be re-
turned but the rewritten model code printed

See Also

`libbi`
**Examples**

```r
bi_object <- libbi(model = system.file(package="rbi", "PZ.bi"))

run(bi_object, client="sample", target="prior")
bi_file_summary(bi_object)
```

---

**Description**

The method `sample` launches `libbi` to sample from a (prior, posterior or joint) distribution. See the options to `run.libbi` for how to specify the various components of sampling with LibBi, and the LibBi manual for all options that can be passed when the client is `sample`.

If `x` is given as a 'bi_model', a `libbi` object will be created from the model For the help page of the base R `sample` function, see `sample`.

**Usage**

```r
## S3 method for class 'libbi'
sample(x, ...)

## S3 method for class 'bi_model'
sample(x, ...)
```

**Arguments**

- `x` a `libbi` or `bi_model` object, or the name of a file containing the model
- `...` options to be passed to `run.libbi`

**Value**

an updated `libbi` object

---

**sample_obs**

*Sample observations from a LibBi model that has been run*

**Description**

Sample observations from a LibBi model that has been run

**Usage**

```r
sample_obs(x, ...)
```
Arguments

x  
a libbi object

...  
any options to pass to LibBi

Value

the original libbi object with added variables in the output file for sampled observations

Author(s)

Sebastian Funk

---

save_libbi  Write results of a LibBi run to an RDS file

Description

This saves all options, files and outputs of a LibBi run to an RDS file specified

Usage

```r
# S3 method for class 'libbi'
save_libbi(x, name, supplement, split = FALSE, ...)
```

Arguments

x  
a libbi object

name  
name of the RDS file(s) to save to. If split=TRUE, this will be taken as a base for the names of the files to be created, e.g. 'dir/name.rds' to create files of the form name_....rds in directory 'dir'.

supplement  
any supplementary data to save

split  
Logical, defaults to FALSE. Should the objects from the LibBi run be saved separately in a folder.

...  
any options to saveRDS

Value

the return value of saveRDS, i.e. NULL invisibly
### set_name

**Set the name of a bi model**

**Description**

Changes the name of a bi model (first line of the .bi file) to the specified name.

**Usage**

```r
## S3 method for class 'bi_model'
set_name(x, name, ...)
```

**Arguments**

- `x`: a `bi_model` object
- `name`: Name of the model
- `...`: ignored

**Value**

the updated `bi_model` object

**See Also**

`bi_model`

**Examples**

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ <- set_name(PZ, "new_PZ")
```

---

### simulate

**Using the LibBi wrapper to simulate**

**Description**

The method `simulate` launches LibBi to simulate a model by passing `target="joint"` to LibBi. If `x` is given as a `bi_model`, a `libbi` object will be created from the model.

**Usage**

```r
## S3 method for class 'libbi'
simulate(x, ...)

## S3 method for class 'bi_model'
simulate(x, ...)
```
Arguments

x a libbi or bi_model object, or the name of a file containing the model

Value

an updated bi_model object

Description

This reads in the output file of the libbi object (which has been run before) and prints summary information of parameters

Usage

## S3 method for class 'libbi'
summary(
  object,
  type = c("param", "state", "noise", "obs"),
  quantiles = c(0.25, 0.75),
  na.rm = FALSE,
  ...
)

Arguments

object a libbi object

type one of "param" (default), "state", "noise" or "obs", the variable type to summarise

quantiles quantiles to calculate (default: quartiles); minimum, median, mean and maximum are always calculated

na.rm logical; if true, any na and NaN’s are removed before calculations are performed

Value

nothing (invisible NULL)
Unequals.bi_model

Description

Ignores differences in the model name.

Usage

## S3 method for class 'bi_model'
e1 != e2, ...

Arguments

e1 a bi_model
e2 a bi_model
...

Value

TRUE or FALSE, depending on whether the models are equal or not

Examples

model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
PZ != PZ # FALSE

update

Update a libbi object

Description

This updates all the time stamps in a libbi object; it is useful after

Usage

## S3 method for class 'libbi'
update(x, ...)

Arguments

x a libbi object
...

Value

a libbi object with updated timestamps
**var_names**

*Get variable names in a LibBi model*

Description

Get variable names of one or more type(s)

This returns all variable names of a certain type ("param", "state", "obs", "noise", "const") contained in the model of a `libbi` object

Usage

```r
var_names(x, vars, type, dim = FALSE, opt = FALSE, aux = FALSE)
```

Arguments

- `x`: a `bi_model` object
- `vars`: a character vector of variable names; if given, only these variables names will be considered
- `type`: a character vector of one or more types
- `dim`: logical; if set to TRUE, names will contain dimensions in brackets
- `opt`: logical; if set to TRUE, names will contain options (e.g., has_output)
- `aux`: logical; if set to TRUE, auxiliary names will be returned

Value

a character vector of variable names

---

**write_model**

*Writes a bi model to a file.*

Description

Writes a bi model to a file given by `filename`. The extension `.bi` will be added if necessary.

Usage

```r
## S3 method for class 'bi_model'
write_model(x, filename, update.name = TRUE, ...)
```
write_model

Arguments

- x: a `bi_model` object, or a `libbi` object containing a model
- filename: name of the file to be written
- update.name: whether to update the model name with the file name
- ...: ignored

Value

the return value of the `writeLines` call.

See Also

`bi_model`

Examples

```r
model_file_name <- system.file(package="rbi", "PZ.bi")
PZ <- bi_model(filename = model_file_name)
write_model(PZ, "PZ.bi")
```
Index

!=.bi_model (Unequals.bi_model), 35
+Topic **package**
  rbi-package, 3
==.bi_model (Equals.bi_model), 13
[.bi_model (Extract.bi_model), 13
[<-bi_model (Extract_assign.bi_model), 14
‘!=.bi_model’ (Unequals.bi_model), 35
‘==.bi_model’ (Equals.bi_model), 13
‘[.bi_model’ (Extract.bi_model), 13
‘[<-bi_model’
  (Extract_assign.bi_model), 14
add_block, 4
attach_data, 4
bi_contents, 6
bi_dim_len, 6
bi_file_summary, 7
bi_generate_dataset, 7
bi_model, 3, 4, 8, 12, 13, 15–21, 26–28, 31, 33–37
bi_open, 6, 7, 9
bi_read, 7, 9, 17, 30
bi_write, 4, 5, 11
enable_outputs, 12
Equals.bi_model, 13
Extract.bi_model, 13
Extract_assign.bi_model, 14
extract_sample, 15
filter, 15, 15, 21, 22, 28
fix, 8, 16, 16, 27
flatten, 17
get_block, 17
get_const, 18
get_dims, 18
get_name, 8, 19
get_traces, 19
insert_lines, 8, 20
join, 21
libbi, 3–6, 8–11, 15, 16, 19–21, 21, 22–25, 28–37
logLik, 22, 22
optimise, 21–23, 23, 24, 28
option_list, 23
option_string, 24
plot, 21
predict, 24
print_log, 25
RBi (rbi-package), 3
rbi (rbi-package), 3
rbi-package, 3
read_libbi, 25
remove_lines, 8, 26
remove_vars, 27
replace_all, 8, 27
rewrite, 21, 22, 28, 28
run, 28
run.libbi, 7, 15, 22, 23, 28, 31, 34
sample, 21, 22, 24, 28, 31, 31
sample.libbi, 7
sample_obs, 31
save_libbi, 32
saveRDS, 32
set_name, 8, 33
simulate, 33
summary, 34
to_input, 27
Unequals.bi_model, 35
update, 35
var_names, 36
INDEX

write_model, 8, 36
writeLines, 37