Package: embr (via r-universe)

November 1, 2024

Title Model Builder Utility Functions and Virtual Classes

Version 0.0.1.9037

Description Utility functions and virtual classes shared by model builder packages such as tmbr, jmbr and smbr.

License MIT + file LICENSE

URL https://github.com/poissonconsulting/embr

BugReports https://github.com/poissonconsulting/embr/issues

Depends R (>= 4.1)

Imports beepr, chk, dplyr, extras, foreach, generics, ggplot2, graphics, lifecycle, lubridate, mcmcdata, mcmcderive (>= 0.1.2.9002), mcmcr (>= 0.6.1.9001), matrixStats, newdata, nlist, plyr, purrr, rescale, rlang, stats, stringr, term, tibble, tidyr, universals, utils

Suggests cli, covr, jmbr, testthat

RdMacros lifecycle

Remotes poissonconsulting/jmbr, poissonconsulting/mcmcr, poissonconsulting/mcmcdata, poissonconsulting/mcmcderive, poissonconsulting/newdata, poissonconsulting/rescale

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Repository https://poissonconsulting.r-universe.dev

RemoteUrl https://github.com/poissonconsulting/embr

RemoteRef HEAD

RemoteSha 3ce9b551d194c6db74ba34f02623fb7fa5fb6df6

Contents

`add_analyses` *Add analyses*

Description

Add analyses

Usage

```
add_analyses(x, x2)
```

Arguments

`x`	An mb_analysis or mb_analyses object.
`x2`	n mb_analysis or mb_analyses object.

Value

An object of class mb_analyses.

`add_models` *Add model(s)*

Description

Add model(s)

Usage

```
add_models(x, x2)
```

Arguments

x	An mb_model or mb_models object.
x2	n mb_model or mb_models object.

Value

An object of class mb_models.

`analyse` *Analyse*

Description

Analyse

Usage

```
analyse(x, ...)
```

Arguments

x	The object to analyse.
...	Additional arguments.

`analyse.character` *Analyse*

Description

Analyse

Usage

```
## S3 method for class 'character'
analyse(
  x,
  data,
  select_data = list(),
  nchains = getOption("mb.nchains", 3L),
  niters = getOption("mb.niters", 1000L),
  nthin = getOption("mb.nthin", 1L),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

x	An object inheriting from class mb_model or a list of such objects.
data	The data frame to analyse.
select_data	A named list specifying the columns to select and their associated classes and values as well as transformations and scaling options.
nchains	A count of the number of chains.
niters	A count of the number of simulations to save per chain.
nthin	A count of the thining interval or NULL (in which case taken from model).
parallel	A flag indicating whether to perform the analysis in parallel if possible.
quiet	A flag indicating whether to disable tracing information.
glance	A flag indicating whether to print a model summary.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

---

analyse.mb_model *Analyse*

Description

Analyse

Usage

```
## S3 method for class 'mb_model'
analyse(
  x,
  data,
  nchains = getOption("mb.nchains", 3L),
  niters = getOption("mb.niters", 1000L),
  nthin = getOption("mb.thin", NULL),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

x	An object inheriting from class mb_model or a list of such objects.
data	The data frame to analyse.
nchains	A count of the number of chains.
niters	A count of the number of simulations to save per chain.

nthin	A count of the thining interval or NULL (in which case taken from model).
parallel	A flag indicating whether to perform the analysis in parallel if possible.
quiet	A flag indicating whether to disable tracing information.
glance	A flag indicating whether to print a model summary.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

`analyse.mb_models` *Analyse*

Description

Analyse

Usage

```
## S3 method for class 'mb_models'
analyse(
  x,
  data,
  nchains = getOption("mb.nchains", 3L),
  niters = getOption("mb.niters", 1000L),
  nthin = getOption("mb.thin", NULL),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

x	An object inheriting from class mb_model or a list of such objects.
data	The data frame to analyse.
nchains	A count of the number of chains.
niters	A count of the number of simulations to save per chain.
nthin	A count of the thining interval or NULL (in which case taken from model).
parallel	A flag indicating whether to perform the analysis in parallel if possible.
quiet	A flag indicating whether to disable tracing information.
glance	A flag indicating whether to print a model summary.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

`analyse1` *Analyse*

Description

Analyse

Usage

```
analyse1(
  model,
  data,
  loaded,
  nchains,
  niters,
  nthin,
  quiet,
  glance,
  parallel,
  ...
)
```

Arguments

`model`	The mb_model to analyse.
`data`	The data.
`loaded`	The loaded model.
`nchains`	chains.
`niters`	iters
`nthin`	thin
`quiet`	quiet
`glance`	glance
`parallel`	parallel
`...`	Additional arguments.

analyses *MB Analyses*

Description

Creates an object inherting from class mb_analyses.

Usage

```
analyses(...)
```

Arguments

...	Named objects.

Value

An object inheriting from class mb_analyses.

analyse_residuals *Analyse Residuals*

Description

Analyse Residuals

Usage

```
analyse_residuals(x)
```

Arguments

x	The mb_analysis object to analyse the residuals for.

as.analyses *Coerce to an mb_analyses object*

Description

Coerce to an mb_analyses object

Usage

```
as.analyses(x, ...)
```

Arguments

x	object to coerce.
...	Unused.

as.model *Coerce to an mb_model object*

Description

Coerce to an mb_model object

Usage

```
as.model(x, ...)
```

Arguments

x	object to coerce.
...	Unused.

as.models *Coerce to an mb_models object*

Description

Coerce to an mb_models object

Usage

```
as.models(x, ...)
```

Arguments

x	object to coerce.
...	Unused.

backwards *Backwards*

Description

Perform backwards step-wise regression on a model. Returns a list of the analysis at each step starting with the full model.

Usage

```
backwards(
  model,
  data,
  drops = list(),
  conf_level = getOption("mb.conf_level", 0.95),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

model	An object.
data	Data.
drops	A list of character vectors specifying the scalar parameters to consider.
conf_level	A number specifying the confidence level. By default 0.95.
beep	A flag indicating whether to beep on completion of the analysis.
...	Unused.

Details

drop is a list of character vectors specifying the scalar parameters to possibly drop. If a list element consists of two or more strings then the earlier strings are only available to drop after the later strings have been eliminated. This allows polynomial dependencies to be respected.

Value

A list of the analyses.

base_model *Base Model*

Description

Base Model

Usage

```
base_model(model, drops = list())
```

Arguments

model	The full model.
drops	A list of character vectors specifying possible drops.

Value

The base model (with all drops).

check_mb_analysis *Check MB Analysis*

Description

Check MB Analysis

Usage

```
check_mb_analysis(object, object_name = substitute(object))
```

Arguments

object	The object to check.
object_name	A string of the object name.

Value

The object or throws an informative error.

check_mb_code *Check MB Code*

Description

Check MB Code

Usage

```
check_mb_code(object, object_name = substitute(object))
```

Arguments

object	The object to check.
object_name	A string of the object name.

Value

The object or throws an informative error.

check_mb_model *Check MB Model*

Description

Check MB Model

Usage

```
check_mb_model(object, object_name = substitute(object))
```

Arguments

object	The object to check.
object_name	A string of the object name.

Value

The object or throws an informative error.

check_model_pars *Check Model Parameters*

Description

Check Model Parameters

Usage

```
check_model_pars(x, fixed, random, derived, drops)
```

Arguments

x	The model code to check.
fixed	A string of a regular expression specifying the fixed parameters to monitor.
random	NULL or a character vector of the random effects.
derived	NULL or a character vector of the derived parameters.
drops	NULL or a character vector of the parameters to drop.

Value

The possibly updated derived parameters.

check_uniquely_named_list *Check Uniquely Named List*

Description

Check Uniquely Named List

Usage

```
check_uniquely_named_list(x, x_name = substitute(x))
```

Arguments

x	The object to check.
x_name	A string of the objects name.

Value

The original object or throws an informative error.

code *Code*

Description

Gets the MB code for an object.

Usage

```
code(object, ...)
```

Arguments

object	The object.
...	Additional arguments.

Value

An object inheriting from class mb_code.

coef.mb_analyses *Coef TMB Analyses*

Description

Coefficients for fixed parameters from an ML based MB analyses averaged by IC weights.

Usage

```
## S3 method for class 'mb_analyses'
coef(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  ...
)
```

Arguments

object	The mb_analyses object.
param_type	A flag specifying whether 'fixed', 'random' or 'derived' terms.
include_constant	A flag specifying whether to include constant terms.
conf_level	A number specifying the confidence level. By default 0.95.
estimate	The function to use to calculate the estimate for Bayesian models.
...	Not used.

Value

A tidy tibble of the coeffcient terms with the model averaged estimate, the Akaike's weight and the proportion of models including the term.

---

coef.mb_analysis *Coef JAGS Analysis*

Description

Coefficients for a JAGS analysis.

Usage

```
## S3 method for class 'mb_analysis'
coef(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  simplify = FALSE,
  ...
)
```

Arguments

object	The mb_analysis object.
param_type	A flag specifying whether 'fixed', 'random' or 'derived' terms.
include_constant	A flag specifying whether to include constant terms.
conf_level	A number specifying the confidence level. By default 0.95.
estimate	The function to use to calculating the estimate for Bayesian models.
simplify	A flag specifying whether to drop sd and zscore columns and return svalue instead of pvalue.
...	Not used.

Details

The zscore is mean / sd.

Value

A tidy tibble of the coefficient terms.

---

coef.mb_meta_analyses *Coef TMB Meta Analyses*

Description

Coef TMB Meta Analyses

Usage

```
## S3 method for class 'mb_meta_analyses'
coef(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  ...
)
```

Arguments

object	The mb_meta_analyses object.
param_type	A flag specifying whether 'fixed', 'random' or 'derived' terms.
include_constant	A flag specifying whether to include constant terms.
conf_level	A number specifying the confidence level. By default 0.95.
estimate	The function to use to calculate the estimate.
...	Not used.

Value

A tidy tibble.

coef.mb_meta_analysis *Coef TMB Meta Analysis*

Description

Coef TMB Meta Analysis

Usage

```
## S3 method for class 'mb_meta_analysis'
coef(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  ...
)
```

Arguments

object	The mb_meta_analysis object.
param_type	A flag specifying whether 'fixed', 'random' or 'derived' terms.
include_constant	A flag specifying whether to include constant terms.
conf_level	A number specifying the confidence level. By default 0.95.
estimate	The function to use to calculate the estimate.
...	Not used.

Value

A tidy tibble.

coef_profile *Coef Profile*

Description

Gets coef confidence limits by likelihood profiling.

Usage

```
coef_profile(object, ...)
```

Arguments

object	The object.
...	Unused.

`coef_profile.mb_analyses` *Coef TMB Analyses*

Description

Coefficients for fixed parameters from an ML based MB analyses averaged by IC weights.

Usage

```
## S3 method for class 'mb_analyses'
coef_profile(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  parallel = getOption("mb.parallel", FALSE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

object	The mb_analyses object.
param_type	A flag specifying whether 'fixed', 'random' or 'derived' terms.
include_constant	A flag specifying whether to include constant terms.
conf_level	A number specifying the confidence level. By default 0.95.
estimate	The function to use to calculate the estimate for Bayesian models.
parallel	A flag indicating whether to using parallel backend provided by foreach.
beep	A flag indicating whether to beep on completion of the analysis.
...	Not used.

Value

A tidy tibble of the coeffcient terms with the model averaged estimate, the Akaike's weight and the proportion of models including the term.

`coef_profile.mb_analysis` *Coef Profile Analysis*

Description

Coefficients for a analysis.

Usage

```
## S3 method for class 'mb_analysis'
coef_profile(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  parallel = getOption("mb.parallel", FALSE),
  beep = getOption("mb.profile", TRUE),
  simplify = TRUE,
  ...
)
```

Arguments

`object`	The mb_analysis object.
`param_type`	A flag specifying whether 'fixed', 'random' or 'derived' terms.
`include_constant`	A flag specifying whether to include constant terms.
`conf_level`	A number specifying the confidence level. By default 0.95.
`estimate`	The function to use to calculating the estimate for Bayesian models.
`parallel`	A flag indicating whether to using parallel backend provided by foreach.
`beep`	A flag indicating whether to beep on completion of the analysis.
`simplify`	A flag specifying whether to simplify with svalue.
`...`	Not used.

Details

The `zscore` is mean / sd.

Value

A tidy tibble of the coefficient terms.

`coef_profile.mb_meta_analyses` *Coef TMB Meta Analyses*

Description

Coef TMB Meta Analyses

Usage

```
## S3 method for class 'mb_meta_analyses'
coef_profile(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  parallel = getOption("mb.parallel", FALSE),
  beep = getOption("mb.parallel", TRUE),
  ...
)
```

Arguments

`object`	The mb_meta_analyses object.
`param_type`	A flag specifying whether 'fixed', 'random' or 'derived' terms.
`include_constant`	A flag specifying whether to include constant terms.
`conf_level`	A number specifying the confidence level. By default 0.95.
`estimate`	The function to use to calculate the estimate.
`parallel`	A flag indicating whether to using parallel backend provided by foreach.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Not used.

Value

A tidy tibble.

`coef_profile.mb_meta_analysis` *Coef TMB Meta Analysis*

Description

Coef TMB Meta Analysis

Usage

```
## S3 method for class 'mb_meta_analysis'
coef_profile(
  object,
  param_type = "fixed",
  include_constant = TRUE,
  conf_level = getOption("mb.conf_level", 0.95),
  estimate = getOption("mb.estimate", median),
  parallel = getOption("mb.parallel", FALSE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

`object`	The mb_meta_analysis object.
`param_type`	A flag specifying whether 'fixed', 'random' or 'derived' terms.
`include_constant`	A flag specifying whether to include constant terms.
`conf_level`	A number specifying the confidence level. By default 0.95.
`estimate`	The function to use to calculate the estimate.
`parallel`	A flag indicating whether to using parallel backend provided by foreach.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Not used.

Value

A tidy tibble.

comment_string *Comment String*

Description

Returns the regular expression used to identify the start of a comment in an mb_code object.

Usage

```
comment_string(object, ...)
```

Arguments

object	The mb code object.
...	Unused.

Value

A string of the regular expression.

data_set *Data Set*

Description

Gets the data set for an object inheriting from class mb_analysis.

Usage

```
data_set(
  x,
  modify = FALSE,
  numericize_factors = FALSE,
  marginalize_random_effects = FALSE,
  ...
)
```

Arguments

x	The object.
modify	A flag indicating whether to modify the data.
numericize_factors	A flag indicating whether to convert factors to integers if modifying the data.
marginalize_random_effects	A flag indicating whether to set each factor in one or more random effects at its first level.
...	Unused.

Value

The data set as a tibble.

`density99` *Density99*

Description

dummy data

Usage

```
density99
```

Format

An object of class `data.frame` with 300 rows and 5 columns.

`drop_pars` *Drop pars*

Description

Drops named scalar fixed pars from an object by fixing them at 0.

Usage

```
drop_pars(x, pars = character(0), ...)
```

Arguments

`x`	The object.
`pars`	A character vector of the pars to drop.
`...`	Not used.

Value

The updated object.

elapsed *Elapsed*

Description

Get elapsed duration.

Usage

```
elapsed(x, ...)
```

Arguments

x	The object to calculate it for.
...	Not used.

estimates.mb_analysis *Estimates*

Description

Calculates the estimates for an MCMC object.

Usage

```
## S3 method for class 'mb_analysis'
estimates(x, param_type = "fixed", ...)
```

Arguments

x	An object.
param_type	A string indicating the type of terms to get the names for.
...	Other arguments passed to methods.

Value

A list of uniquely named numeric objects.

See Also

Other MCMC manipulations: bind_chains(), bind_iterations(), collapse_chains(), split_chains()

fitted.mb_analysis *Fitted Values*

Description

Extract fitted values for a MB analysis.

Usage

```
## S3 method for class 'mb_analysis'
fitted(object, ...)
```

Arguments

object	The MB analysis object.
...	Unused.

Details

The new_expr in the model must include the term 'fit'.

Value

The analysis data set with the fitted values.

get_analysis_mode *Get Analysis Mode*

Description

Gets analysis mode.

Usage

```
get_analysis_mode()
```

Details

Retrieves what is set for each of the following package options.

mb.nchains A count of the number of chains.

mb.niters A count of the number of simulations to save per chain.

mb.nthin A count of the thining interval.

mb.parallel A flag indicating whether to perform the analysis in parallel.

mb.quiet A flag indicating whether to disable tracing information.

mb.beep A flag indicating whether to beep on completion of the analysis.

mb.glance A flag indicating whether to print a model summary.

mb.nreanalyses A count specifying the maximum number of reanalyses.

mb.rhat A number specifying the rhat threshold.

mb.esr A number specifying the minimum effective sampling rate.

mb.duration The maximum total time to spend on analysis and reanalysis.

mb.conf_level A number specifying the confidence level.

Value

A named list of the current package options.

Examples

```
## Not run:
get_analysis_mode()

## End(Not run)
```

`get_model` *Retrieve model*

Description

Constucts a new analysis object

Usage

```
get_model(analysis)
```

Arguments

`analysis`	An object of class ‘"mb_analysis"‘.

IC *Information Criterion*

Description

Calculates Information Criterion for an analysis.

Usage

```
IC(object, ...)
```

Arguments

object	The object to calculate the IC for.
...	Not used.

Value

The Information Criteron as a number.

is.lmb_analysis *Is a LMB Analysis*

Description

Tests whether x is an object of class 'lmb_analysis'

Usage

```
is.lmb_analysis(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.lmb_code *Is a LMB Code*

Description

Tests whether x is an object of class 'lmb_code'

Usage

```
is.lmb_code(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.lmb_model *Is a LMB Model*

Description

Tests whether x is an object of class 'lmb_model'

Usage

```
is.lmb_model(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.mb_analyses *Is MB Analyses?*

Description

Tests whether x is an object of class 'mb_analyses'

Usage

```
is.mb_analyses(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.mb_analysis *Is MB Analysis?*

Description

Tests whether x is an object of class 'mb_analysis'

Usage

```
is.mb_analysis(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.mb_code *Is MB Code?*

Description

Tests whether x is an object of class 'mb_model'

Usage

```
is.mb_code(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.mb_model *Is MB Model?*

Description

Tests whether x is an object of class 'mb_model'

Usage

```
is.mb_model(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.mb_models *Is MB Models?*

Description

Tests whether x is an object of class 'mb_models'

Usage

```
is.mb_models(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.mb_null_analysis *Is MB Null Analysis?*

Description

Tests whether x is an object of class 'mb_null_analysis'

Usage

```
is.mb_null_analysis(x)
```

Arguments

x	The object to test.

Value

A flag indicating whether the test was positive.

is.syntactic *Is syntactic*

Description

Is syntactic

Usage

```
is.syntactic(x)
```

Arguments

x	A character of possible variable names.

Value

A logical vector indicating whether a syntactically correct variable name.

Examples

```
is.syntactic(c("0", "x", "1x", "x y", "x1"))
```

is_bayesian *Test if is bayesian*

Description

Test if is bayesian

Usage

```
is_bayesian(x, ...)
```

Arguments

x	the object.
...	Unused

Value

A flag indicating wether bayesian.

is_frequentist *Test if is frequentist*

Description

Test if is frequentist

Usage

```
is_frequentist(x)
```

Arguments

x	the object.

Value

A flag indicating whether frequentist

is_namedlist *Is Named List*

Description

Is Named List

Usage

```
is_namedlist(x)
```

Arguments

x	The object to test.

Value

A flag.

Examples

```
is_namedlist(1)
is_namedlist(list())
is_namedlist(list(1))
is_namedlist(list(x = 1))
is_namedlist(list(x = list(y = 2)))
```

load_model *Load Model*

Description

Load Model

Usage

```
load_model(x, quiet, ...)
```

Arguments

x	The model to load.
quiet	A flag indicating whether to suppress warnings and output.
...	Additional arguments.

logLik.mb_analysis *Log-Likelihood*

Description

Log-Likelihood for a MB analysis.

Usage

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

Arguments

object	The mb_analysis object.
...	unused.

logLik.mb_null_analysis *Log-Likelihood*

Description

Log-Likelihood for a MB NULL analysis.

Usage

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

Arguments

object	The mb_analysis object.
...	unused.

make_all_models *Make All Models*

Description

Make All Models

Usage

```
make_all_models(model, drops = list())
```

Arguments

model	The full model.
drops	A list of character vectors specifying possible drops.

Value

A list of objects inheriting from class mb_model.

mb_code *MB Code*

Description

Identifies the type of the code and creates an object of the appropriate class.

Usage

```
mb_code(template)

new_mb_code(x, class)
```

Arguments

template	A string, a braced “ expression (unquoted or quoted), or an object of class ‘"mb_code"‘.
x	A string or a braced “ expression.
class	The class of the new object.

Value

An object inheriting from class mb_code.

Examples

```
x <- mb_code(
  "#include <TMB.hpp>

template<class Type>

Type objective_function<Type>::operator() () {
DATA_VECTOR(Count);
PARAMETER(bIntercept);

int n = Count.size();

Type nll = 0.0;
for(int i = 0; i < n; i++){
  nll -= dpois(Count(i), bIntercept, true);
}
return nll;
}
"
)
class(x)
```

`mcmc_derive.mb_analyses` *Derive*

Description

Calculate derived parameters.

Usage

```
## S3 method for class 'mb_analyses'
mcmc_derive(
  object,
  new_data = data_set(object),
  new_expr = NULL,
  new_values = list(),
  term = "prediction",
  modify_new_data = NULL,
  ref_data = FALSE,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

`object`	An object inheriting from class mb_analysis.
`new_data`	The data frame to calculate the predictions for.
`new_expr`	A string of R code specifying the predictive relationship.
`new_values`	A named list of new or replacement values to pass to new_expr.
`term`	A string of the term in new_expr.
`modify_new_data`	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
`ref_data`	A flag or a data frame with 1 row indicating the reference values for calculating the effects size.
`new_expr_vec`	A flag specifying whether to vectorize the new_expr code.
`parallel`	A flag indicating whether to do predictions using parallel backend provided by foreach.
`quiet`	A flag indicating whether to disable tracing information.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Additional arguments.

Value

A object of class mcmcr.

`mcmc_derive.mb_analysis` *Derive*

Description

Calculate derived parameters.

Usage

```
## S3 method for class 'mb_analysis'
mcmc_derive(
  object,
  new_data = data_set(object),
  new_expr = NULL,
  new_values = list(),
  term = "prediction",
  modify_new_data = NULL,
  ref_data = FALSE,
  ref_fun2 = proportional_change2,
  random_effects = NULL,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

`object`	An object inheriting from class mb_analysis.
`new_data`	The data frame to calculate the predictions for.
`new_expr`	A string of R code specifying the predictive relationship.
`new_values`	A named list of new or replacement values to pass to new_expr.
`term`	A string of the term in new_expr.
`modify_new_data`	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
`ref_data`	A flag or a data frame with 1 row indicating the reference values for calculating the effects size.
`ref_fun2`	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.

random_effects	A named list specifying the random effects and the associated factors.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
parallel	A flag indicating whether to do predictions using parallel backend provided by foreach.
quiet	A flag indicating whether to disable tracing information.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

Value

A object of class mcmcr.

`mcmc_derive_data.mb_analyses` *Derive Data*

Description

Calculate derived parameters.

Usage

```
## S3 method for class 'mb_analyses'
mcmc_derive_data(
  object,
  new_data = data_set(object),
  new_expr = NULL,
  new_values = list(),
  term = "prediction",
  modify_new_data = NULL,
  ref_data = FALSE,
  ref_fun2 = proportional_change2,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

object	An object inheriting from class mb_analysis.
new_data	The data frame to calculate the predictions for.
new_expr	A string of R code specifying the predictive relationship.
new_values	A named list of new or replacement values to pass to new_expr.

term	A string of the term in new_expr.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
ref_data	A flag or a data frame with 1 row indicating the reference values for calculating the effects size.
ref_fun2	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
parallel	A flag indicating whether to do predictions using parallel backend provided by foreach.
quiet	A flag indicating whether to disable tracing information.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

Value

A object of class mcmc_data.

`mcmc_derive_data.mb_analysis` *Derive Data*

Description

Calculate derived parameters.

Usage

```
## S3 method for class 'mb_analysis'
mcmc_derive_data(
  object,
  new_data = data_set(object),
  new_expr = NULL,
  new_values = list(),
  term = "prediction",
  modify_new_data = NULL,
  ref_data = FALSE,
  ref_fun2 = proportional_change2,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  random_effects = NULL,
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

object	An object inheriting from class mb_analysis.
new_data	The data frame to calculate the predictions for.
new_expr	A string of R code specifying the predictive relationship.
new_values	A named list of new or replacement values to pass to new_expr.
term	A string of the term in new_expr.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
ref_data	A flag or a data frame with 1 row indicating the reference values for calculating the effects size.
ref_fun2	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
random_effects	A named list specifying the random effects and the associated factors.
parallel	A flag indicating whether to do predictions using parallel backend provided by foreach.
quiet	A flag indicating whether to disable tracing information.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

Value

A object of class mcmc_data.

---

model *MB Model*

Description

Creates MB model.

Usage

```
model(
  x = NULL,
  ...,
  code = NULL,
  gen_inits = NULL,
  random_effects = list(),
  fixed = getOption("mb.fixed", "^[^e]"),
  derived = character(0),
  select_data = list(),
```

```
  center = character(0),
  scale = character(0),
  modify_data = identity,
  nthin = getOption("mb.nthin", 1L),
  new_expr = NULL,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  modify_new_data = identity,
  drops = list()
)
```

Arguments

x	A string, or an object inheriting from class ‘"mb_code"‘.
...	Unused arguments.
code	Passed on to [mb_code()]. If ‘x‘ is not ‘NULL‘, ‘code‘ must be ‘NULL‘, and vice versa.
gen_inits	A single argument function taking the modified data and returning a named list of initial values.
random_effects	A named list specifying the random effects and the associated factors.
fixed	A string of a regular expression specifying the fixed pars to monitor.
derived	A character vector of the derived pars to monitor.
select_data	A named list specifying the columns to select and their associated classes and values as well as transformations and scaling options.
center	A character vector of the columns to center.
scale	A character vector of the columns to scale (after centering).
modify_data	A single argument function to modify the data (in list form) immediately prior to the analysis.
nthin	A count specifying the thinning interval.
new_expr	A string of R code specifying the predictive relationships.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
drops	A list of character vector of possible scalar pars to drop (fix at 0).

Details

For tmb models gen_inits must specify all the fixed pars. Missing random pars are assigned the value 0.

For jmb models unspecified the initial values for each chain are drawn from the prior distributions.

Value

An object inherting from class mb_model.

See Also

check_data rescale_c

`models` *MB Models*

Description

Creates an object inherting from class mb_models.

Usage

```
models(...)
```

Arguments

`...`	Named objects.

Value

An object inheriting from class mb_models.

`modify_data` *Modify Data*

Description

Modifies a data frame to the form it will be passed to the analysis code.

Usage

```
modify_data(data, model, numericize_factors = FALSE)
```

Arguments

`data`	The data to modify.
`model`	An object inheriting from class mb_model.
`numericize_factors`	A flag indicating whether to convert factors to integer.

Value

The modified data in list form.

modify_new_data *Modify New Data*

Description

Modifies a data frame to the form it will be passed to the analysis code.

Usage

```
modify_new_data(
  data,
  data2,
  model,
  modify_new_data = NULL,
  numericize_factors = FALSE
)
```

Arguments

data	The data to modify.
data2	The base data.
model	An object inheriting from class mb_model.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
numericize_factors	A flag indicating whether to convert factors to integer.

Value

The modified data in list form.

monitor *pars to monitor*

Description

pars to monitor

Usage

```
monitor(object, param_type = "all")
```

Arguments

object	An mb model object to get the pars for.
param_type	A string specifying the type of pars to get.

Value

A character vector of the pars to monitor.

---

new_analysis *MB Analysis*

Description

Constucts a new analysis object

Usage

```
new_analysis(x, class)
```

Arguments

x	A list.
class	The class of the new object.

---

new_expr *new_expr*

Description

new_expr

Usage

```
new_expr(object, ...)
```

Arguments

object	The object to get for.
...	Not used.

Value

The new_expr

new_expr<- *new_expr set*

Description

new_expr set

Usage

```
new_expr(object) <- value
```

Arguments

object	The object to set for.
value	The new value of new expr.

Value

The modified object.

ngens *Total Number of MCMC simulations generated (including warmup)*

Description

Total Number of MCMC simulations generated (including warmup)

Usage

```
ngens(x, ...)
```

Arguments

x	The object
...	Unused.

Value

A count.

nmodels *Number Models*

Description

Number Models

Usage

```
nmodels(x, ...)
```

Arguments

x	the object.
...	Named objects.

Value

An integer of the number of models

nterms.mb_analysis *Number of terms*

Description

Number of terms

Usage

```
## S3 method for class 'mb_analysis'
nterms(x, param_type = "fixed", include_constant = TRUE, ...)
```

Arguments

x	The object to get the nterms for
param_type	A string indicating the type of terms to get the names for.
include_constant	A flag specifying whether to include constant terms.
...	unused

nthin *Thinning Rate*

Description

Thinning Rate

Usage

```
nthin(x, ...)
```

Arguments

x	The object
...	Unused.

Value

A count.

params *Parameter Descriptions*

Description

Parameter Descriptions

Arguments

param_type	A string indicating the type of terms to get the names for.
type	A string of the residual type.
...	Unused.

plot.mb_analysis *Plot Analysis*

Description

Plot Analysis

Usage

```
## S3 method for class 'mb_analysis'
plot(x, param_type = "fixed", ...)
```

Arguments

x	The analysis object to plot
param_type	A string indicating the type of terms to get the names for.
...	Unused.

plot_data *Plot Data*

Description

Plot Data

Usage

```
plot_data(x, ...)
```

Arguments

x	The object to plot the data for.
...	Unused.

plot_residuals *Plot Residuals*

Description

Plot Residuals

Usage

```
plot_residuals(x, ...)
```

Arguments

x	The object to plot the residuals for.
...	Unused.

posterior_predictive_check *Posterior Predictive Check*

Description

Simulates

Usage

```
posterior_predictive_check(x, ...)
```

Arguments

x	The object
...	Unused.

Value

A tibble of the checks.

posterior_predictive_check.mb_analysis *Posterior Predictive Check*

Description

Posterior Predictive Check

Usage

```
## S3 method for class 'mb_analysis'
posterior_predictive_check(x, zeros = TRUE, ...)
```

Arguments

x	The MB analysis object.
zeros	A flag specifying whether to perform a posterior predictive check on the number of zeros in the data.
...	Unused.

Value

A tibble of the checks.

predict.mb_analyses *Predict*

Description

Predict

Usage

```
## S3 method for class 'mb_analyses'
predict(
  object,
  new_data = data_set(object),
  new_expr = NULL,
  new_values = list(),
  term = "prediction",
  conf_level = getOption("mb.conf_level", 0.95),
  modify_new_data = NULL,
  ref_data = FALSE,
  ref_fun2 = proportional_change2,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
```

```
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

object	An object inheriting from class mb_analysis.
new_data	The data frame to calculate the predictions for.
new_expr	A string of R code specifying the predictive relationship.
new_values	A named list of new or replacement values to pass to new_expr.
term	A string of the term in new_expr.
conf_level	A number specifying the confidence level. By default 0.95.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
ref_data	A flag or a data frame with 1 row indicating the reference values for calculating the effects size.
ref_fun2	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
parallel	A flag indicating whether to do predictions using parallel backend provided by foreach.
quiet	A flag indicating whether to disable tracing information.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

---

predict.mb_analysis *Predict*

Description

Predict

Usage

```
## S3 method for class 'mb_analysis'
predict(
  object,
  new_data = data_set(object),
  new_expr = NULL,
  new_values = list(),
```

```
  term = "prediction",
  conf_level = getOption("mb.conf_level", 0.95),
  modify_new_data = NULL,
  ref_data = FALSE,
  ref_fun2 = proportional_change2,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  random_effects = NULL,
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

object	An object inheriting from class mb_analysis.
new_data	The data frame to calculate the predictions for.
new_expr	A string of R code specifying the predictive relationship.
new_values	A named list of new or replacement values to pass to new_expr.
term	A string of the term in new_expr.
conf_level	A number specifying the confidence level. By default 0.95.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
ref_data	A flag or a data frame with 1 row indicating the reference values for calculating the effects size.
ref_fun2	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
random_effects	A named list specifying the random effects and the associated factors.
parallel	A flag indicating whether to do predictions using parallel backend provided by foreach.
quiet	A flag indicating whether to disable tracing information.
beep	A flag indicating whether to beep on completion of the analysis.
...	Additional arguments.

R2 *R2*

Description

Gets the R2 value for an object.

Usage

```
R2(object, ...)
```

Arguments

object	The object.
...	Unused.

Value

An index of the R2 value.

R2.mb_analysis *R2*

Description

Gets the conditional (or marginal) R2 value for the 'response' for an mb_analysis object.

Usage

```
## S3 method for class 'mb_analysis'
R2(
  object,
  response,
  marginal = FALSE,
  term = "prediction",
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  beep = getOption("mb.beep", FALSE),
  ...
)
```

Arguments

object	The object.
response	A string specifying the column in the data corresponding to the response.
marginal	A flag indicating whether to calculate the marginal or conditional R2 value.
term	A string of the term in new_expr.
parallel	A flag indicating whether to do predictions using parallel backend provided by foreach.
quiet	A flag indicating whether to disable tracing information.
beep	A flag indicating whether to beep on completion of the analysis.
...	Unused

Details

The conditional R2 value is the proportion of the variance in the response predicted by the full model. The marginal R2 values is just for the fixed effects ie after marginalizing out the random effects.

Value

A number of the R2 value.

---

random_effects *Random Effects*

Description

Gets the random effects definitions for an object inheriting from class mb_model or mb_analysis.

Usage

```
random_effects(object, ...)
```

Arguments

object	The object.
...	Unused.

Value

The random effects as a sorted named list.

---

reanalyse *Reanalyse*

Description

Reanalyse an analysis.

Usage

```
reanalyse(object, ...)
```

Arguments

object	The object to reanalyse.
...	Additional arguments.

`reanalyse.mb_analyses` *Reanalyse*

Description

Reanalyse

Usage

```
## S3 method for class 'mb_analyses'
reanalyse(
  object,
  rhat = getOption("mb.rhat", 1.1),
  esr = getOption("mb.esr", 0.33),
  nreanalyses = getOption("mb.nreanalyses", 1L),
  duration = getOption("mb.duration", dhours(1)),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

`object`	The object to reanalyse.
`rhat`	A number specifying the rhat threshold.
`esr`	A number specifying the effective sampling rate.
`nreanalyses`	A count between 1 and 7 specifying the maximum number of reanalyses.
`duration`	The maximum total time to spend on analysis/reanalysis.
`parallel`	A flag indicating whether to perform the analysis in parallel if possible
`quiet`	A flag indicating whether to disable tracing information.
`glance`	A flag indicating whether to print summary of model.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Unused arguments.

`reanalyse.mb_analysis` *Reanalyse*

Description

Reanalyse

Usage

```
## S3 method for class 'mb_analysis'
reanalyse(
  object,
  rhat = getOption("mb.rhat", 1.1),
  esr = getOption("mb.esr", 0.33),
  nreanalyses = getOption("mb.nreanalyses", 1L),
  duration = getOption("mb.duration", dhours(1)),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

`object`	The object to reanalyse.
`rhat`	A number specifying the rhat threshold.
`esr`	A number specifying the minimum effective sampling rate.
`nreanalyses`	A count between 0 and 4 specifying the maximum number of reanalyses.
`duration`	The maximum total time to spend on analysis and reanalysis.
`parallel`	A flag indicating whether to perform the analysis in parallel if possible.
`quiet`	A flag indicating whether to disable tracing information.
`glance`	A flag indicating whether to print summary of model.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Unused arguments.

`reanalyse.mb_meta_analyses` *Reanalyse*

Description

Reanalyse

Usage

```
## S3 method for class 'mb_meta_analyses'
reanalyse(
  object,
  rhat = getOption("mb.rhat", 1.1),
  esr = getOption("mb.esr", 0.33),
  nreanalyses = getOption("mb.nreanalyses", 1L),
  duration = getOption("mb.duration", dhours(1)),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

`object`	The object to reanalyse.
`rhat`	A number specifying the rhat threshold.
`esr`	A number specifying the effective sampling rate.
`nreanalyses`	A count between 1 and 7 specifying the maximum number of reanalyses.
`duration`	The maximum total time to spend on analysis/reanalysis.
`parallel`	A flag indicating whether to perform the analysis in parallel if possible
`quiet`	A flag indicating whether to disable tracing information.
`glance`	A flag indicating whether to print summary of model.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Unused arguments.

`reanalyse.mb_meta_analysis` *Reanalyse*

Description

Reanalyse

Usage

```
## S3 method for class 'mb_meta_analysis'
reanalyse(
  object,
  rhat = getOption("mb.rhat", 1.1),
  esr = getOption("mb.esr", 0.33),
  nreanalyses = getOption("mb.nreanalyses", 1L),
  duration = getOption("mb.duration", dhours(1)),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

`object`	The object to reanalyse.
`rhat`	A number specifying the rhat threshold.
`esr`	A number specifying the effective sampling rate.
`nreanalyses`	A count between 1 and 7 specifying the maximum number of reanalyses.
`duration`	The maximum total time to spend on analysis/reanalysis.
`parallel`	A flag indicating whether to perform the analysis in parallel if possible
`quiet`	A flag indicating whether to disable tracing information.
`glance`	A flag indicating whether to print summary of model.
`beep`	A flag indicating whether to beep on completion of the analysis.
`...`	Unused arguments.

reanalyse1 *Reanalyse*

Description

Reanalyse an analysis. For user to override.

Usage

```
reanalyse1(object, parallel, quiet, ...)
```

Arguments

object	The object to reanalyse.
parallel	A flag indicating whether to perform the reanalysis in parallel.
quiet	A flag indicating whether to capture output.
...	Additional arguments.

residuals.mb_analysis *Residuals*

Description

Extract residual values for an MB analysis.

Usage

```
## S3 method for class 'mb_analysis'
residuals(object, type = NULL, ...)
```

Arguments

object	The MB analysis object.
type	A string of the residual type.
...	Unused.

Details

The new_expr in the model must include the term 'residual'.

Value

The analysis data set with the residual values.

rm_comments *Removes comments from the template strings(s) of an mb object or a character vector.*

Description

Removes comments from the template strings(s) of an mb object or a character vector.

Usage

```
rm_comments(object, ...)
```

Arguments

object	The mb object or character vector.
...	Unused.

Value

The mb object or character vector without comments in its template string(s).

sample_size *Sample Size*

Description

Get sample size.

Usage

```
sample_size(object, ...)
```

Arguments

object	The object to calculate the sample size for.
...	Not used.

Value

A count of the sample size.

scalar_nlist *Scalar Named List*

Description

Filters a named list so only scalar elements remain.wiby its names.

Usage

```
scalar_nlist(x)
```

Arguments

x	The named list to sort.

Value

The sorted named list.

Examples

```
scalar_nlist(list(y = 2, x = 1, a = c(10, 1)))
```

sd_priors_by *Multiply Standard Deviation of Priors By*

Description

Multiply Standard Deviation of Priors By

Usage

```
sd_priors_by(x, by = 10, distributions = c("normal", "lognormal", "t"), ...)

## S3 method for class 'mb_model'
sd_priors_by(x, by = 10, distributions = c("normal", "lognormal", "t"), ...)

## S3 method for class 'mb_analysis'
sd_priors_by(
  x,
  by = 10,
  distributions = c("normal", "lognormal", "t"),
  parallel = getOption("mb.parallel", FALSE),
  quiet = getOption("mb.quiet", TRUE),
  glance = getOption("mb.glance", TRUE),
  beep = getOption("mb.beep", TRUE),
  ...
)
```

Arguments

x	The object.
by	A double scalar of the multiplier.
distributions	A character vector of the distributions to adjust. Possible values are "laplace" (double exponential), "logistic", "lognormal", "normal", "t" and "nt" (non-central Student t).
...	Not used.
parallel	A flag indicating whether to perform the analysis in parallel if possible.
quiet	A flag indicating whether to disable tracing information.
glance	A flag indicating whether to print a model summary.
beep	A flag indicating whether to beep on completion of the analysis.

Value

The updated object.

Methods (by class)

- `sd_priors_by(mb_model)`: Multiply Standard Deviation of Priors for an MB model
- `sd_priors_by(mb_analysis)`: Multiply Standard Deviation of Priors for an MB analysis

---

`select_rescale_data` *Select and Rescale Data*

Description

Selects and rescales data.

Usage

```
select_rescale_data(data, model, data2 = data)
```

Arguments

data	The data to modify.
model	An object inheriting from class mb_model.
data2	The base data.

Value

The modified data in list form.

`set_analysis_mode` *Set Analysis Mode*

Description

Sets analysis mode.

Usage

```
set_analysis_mode(mode = "report")
```

Arguments

`mode`	A string of the analysis mode.

Details

The possible modes are as follows:

'debug' To rapidly identify problems with a model definition.

'quick' To quickly test code runs.

'report' To produce results for a report.

'paper' To produce results for a peer-reviewed paper.

'check' To run when checking a package.

'reset' To reset all the options to NULL so that they are the default values for each function call.

In each case the mode is a unique combination of the following package options

mb.nchains A count of the number of chains.

mb.niters A count of the number of simulations to save per chain.

mb.nthin A count of the thining interval.

mb.parallel A flag indicating whether to perform the analysis in parallel.

mb.quiet A flag indicating whether to disable tracing information.

mb.beep A flag indicating whether to beep on completion of the analysis.

mb.glance A flag indicating whether to print a model summary.

mb.nreanalyses A count specifying the maximum number of reanalyses.

mb.rhat A number specifying the rhat threshold.

mb.esr A number specifying the minimum effective sampling rate.

mb.duration The maximum total time to spend on analysis and reanalysis.

mb.conf_level A number specifying the confidence level.

Value

The old options.

Examples

```
## Not run:
set_analysis_mode("reset")

## End(Not run)
```

`simulate_residuals` *Simulate Residuals*

Description

Requires that new_expr includes 'residual <- res_bern(' or 'residual[i] <- res_norm('.

Usage

```
simulate_residuals(x, type = NULL)
```

Arguments

`x`	The MB analysis object.
`type`	A string of the residual type.

Value

An mcmc_data of the simulated residuals.

See Also

extras::res_binom

`sort_by_ic` *Sort Analyses by IC*

Description

Sort Analyses by IC

Usage

```
sort_by_ic(x, ...)
```

Arguments

`x`	the object.
`...`	Unused

Value

The sorted object

`sort_nlist` *Sort Named List*

Description

Sorts a named list by its names.

Usage

```
sort_nlist(x)
```

Arguments

`x`	The named list to sort.

Value

The sorted named list.

Examples

```
sort_nlist(list(y = 2, x = 1, a = 10))
```

`template` *Template*

Description

Gets the template string for an object.

Usage

```
template(object, ...)
```

Arguments

`object`	The object.
`...`	Additional arguments.

Value

The template model code as a string.

template<- *Set Template*

Description

Sets the template for an object.

Usage

```
template(object) <- value
```

Arguments

object	The object.
value	A string of the new template

terms.mb_analysis *Terms*

Description

terms

Usage

```
## S3 method for class 'mb_analysis'
terms(x, param_type = "fixed", include_constant = TRUE, ...)
```

Arguments

x	The mb_analysis object.
param_type	A string indicating the type of terms to get the names for.
include_constant	A flag specifying whether to include constant terms.
...	Not used.

update_model *Update MB Model*

Description

Updates an object inherting from class mb_model.

Usage

```
update_model(
  model,
  code = NULL,
  gen_inits = NULL,
  random_effects = NULL,
  fixed = NULL,
  derived = NULL,
  select_data = NULL,
  center = NULL,
  scale = NULL,
  modify_data = NULL,
  nthin = NULL,
  new_expr = NULL,
  new_expr_vec = getOption("mb.new_expr_vec", FALSE),
  modify_new_data = NULL,
  drops = NULL,
  ...
)
```

Arguments

model	The model to update.
code	A string of the model template or an object inheriting from class mb_code.
gen_inits	A single argument function taking the modified data and returning a named list of initial values.
random_effects	A named list specifying the random effects and the associated factors.
fixed	A string of a regular expression specifying the fixed pars to monitor.
derived	A character vector of the derived pars to monitor.
select_data	A named list specifying the columns to select and their associated classes and values as well as transformations and scaling options.
center	A character vector of the columns to center.
scale	A character vector of the columns to scale (after centering).
modify_data	A single argument function to modify the data (in list form) immediately prior to the analysis.
nthin	A count specifying the thinning interval.

new_expr	A string of R code specifying the predictive relationships.
new_expr_vec	A flag specifying whether to vectorize the new_expr code.
modify_new_data	A single argument function to modify new data (in list form) immediately prior to calculating new_expr.
drops	A list of character vector of possible scalar pars to drop (fix at 0).
...	Unused arguments.

Value

An object inheriting from class mb_model.

Index