

Package: embr (via r-universe)

October 15, 2026

Title Model Builder Utility Functions and Virtual Classes

Version 1.1.0.9004

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 beeper, chk, dplyr, extras, foreach, generics, ggplot2, graphics, lifecycle, lubridate, mcmcdata, mcmcderive (>= 0.1.2.9002), memcr (>= 0.6.2.9002), matrixStats, newdata, nlist, plyr, posterior, priorsense, purrr, rescale, rlang, stats, stringr, term, tibble, tidyr, universals, utils

Suggests cli, covr, jmbr, rmarkdown, scales, smbr2, testthat (>= 3.0.0)

RdMacros lifecycle

Remotes poissonconsulting/extras, poissonconsulting/jmbr, poissonconsulting/memcr, poissonconsulting/mcmcdata, poissonconsulting/mcmcderive, poissonconsulting/newdata, poissonconsulting/rescale, poissonconsulting/smbr2

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Config/Needs/website poissonconsulting/poissontemplate

Config/roxygen2/version 8.1.0.9000

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-10-15 07:24:26 UTC

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

RemoteRef HEAD

RemoteSha fb999dcf2769cf28e29fbd43b63f0ffb33e19ad3

Contents

add_analyses	4
add_models	5
add_sensitivity	5
analyse	6
analyse.character	9
analyse.mb_models	11
analyse_residuals	12
analyses	13
as.analyses	13
as.model	14
as.models	14
backwards	15
base_model	16
check_mb_analysis	16
check_mb_code	17
check_mb_model	17
check_model_pars	18
check_uniquely_named_list	18
code	19
coef.mb_analyses	19
coef.mb_analysis	20
coef.mb_meta_analyses	21
coef.mb_meta_analysis	22
coef_profile	22
coef_profile.mb_analyses	23
coef_profile.mb_analysis	24
coef_profile.mb_meta_analyses	25
coef_profile.mb_meta_analysis	26
comment_string	27
data_set	27
density99	28
drop_pars	28
elapsed	29
estimates.mb_analysis	29
fitted.mb_analysis	30
get_analysis_mode	30
get_model	31
IC	32
is.lmb_analysis	32
is.lmb_code	33
is.lmb_model	33

is.mb_analyses	34
is.mb_analysis	34
is.mb_code	35
is.mb_model	35
is.mb_models	36
is.mb_null_analysis	36
is.syntactic	37
is_bayesian	37
is_frequentist	38
is_namedlist	38
load_model	39
log_lik_draws.mb_analysis	39
log_prior_draws.mb_analysis	40
logLik.mb_analysis	40
logLik.mb_null_analysis	41
make_all_models	41
mb_code	42
mcmc_derive.mb_analysis	43
mcmc_derive_data.mb_analyses	45
mcmc_derive_data.mb_analysis	46
model	48
models	52
modify_data	52
modify_new_data	53
monitor	53
new_analysis	54
new_expr	54
new_expr<-	55
ngens	55
nmodels	56
nterms.mb_analysis	56
nthin	57
params	57
plot.mb_analysis	58
plot_data	58
plot_residuals	59
posterior_predictive_check	59
posterior_predictive_check.mb_analysis	60
powerscale_sensitivity.mb_analysis	60
predict.mb_analyses	61
predict.mb_analysis	62
R2	65
R2.mb_analysis	65
random_effects	66
reanalyse	67
reanalyse.mb_analyses	67
reanalyse.mb_analysis	68
reanalyse.mb_meta_analyses	69

reanalyse.mb_meta_analysis	70
reanalyse1	71
residuals.mb_analysis	71
rm_comments	72
sample_size	72
scalar_nlist	73
sd_priors_by	73
select_rescale_data	74
sensitivity	75
set_analysis_mode	75
simulate_residuals	77
sort_by_ic	77
sort_nlist	78
template	78
template<-	79
terms.mb_analysis	79
update_model	80
Index	82

add_analyses	<i>Add analyses</i>
--------------	---------------------

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	<i>Add model(s)</i>
------------	---------------------

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.

add_sensitivity	<i>Add sensitivity data frame to an analysis object</i>
-----------------	---

Description

Computes power-scaling sensitivity and stores the result as x\$sensitivity. Subsequent calls return x unchanged unless replace = TRUE.

Usage

```
add_sensitivity(x, new_expr = NULL, replace = FALSE, ...)
```

Arguments

x	The object.
new_expr	A string of R code specifying the predictive relationship. Must use replace = TRUE for this to be applied.
replace	A flag specifying whether to replace an existing sensitivity data frame.
...	Arguments passed to powerscale_sensitivity() .

Value

The analysis object with an added sensitivity data frame.

analyse

*Analyse Models***Description**

Fit a Bayesian hierarchical model to data using Stan (via `rstan` or `cmdstanr`) or JAGS (via `rjags`). The estimation backend is determined by the model class (JAGS or Stan) and, for Stan models, by the `stan_engine` argument.

Usage

```
analyse(x, ...)

## 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),
  seed = sample.int(.Machine$integer.max, 1),
  stan_engine = getOption("mb.stan_engine", character(0)),
  niters_warmup = niters,
  ...
)
```

Arguments

<code>x</code>	An <code>mb_model</code> object.
<code>...</code>	Additional arguments passed to the underlying estimation function.
<code>data</code>	A data frame, or a named list of data frames for meta-analysis across datasets.
<code>nchains</code>	A count of the number of chains (default: 3).
<code>niters</code>	A count of the number of iterations to save per chain (default: 1000).
<code>nthin</code>	A count of the thinning interval.
<code>parallel</code>	A flag indicating whether to perform the analysis in parallel if possible.
<code>quiet</code>	A flag indicating whether to disable messages and warnings, including sampling progress.
<code>glance</code>	A flag indicating whether to print a model summary.
<code>beep</code>	A flag indicating whether to beep on completion of the analysis.

seed	A positive whole number specifying the seed to use. The default is random. This is currently only implemented for Stan models.
stan_engine	A string selecting the Stan engine: • "cmdstan-mcmc": MCMC via <code>cmdstanr::sample()</code>. • "cmdstan-pathfinder": pathfinder via <code>cmdstanr::pathfinder()</code>. • "cmdstan-variational": variational ADVI via <code>cmdstanr::variational()</code>. • "cmdstan-optimize": optimization via <code>cmdstanr::optimize()</code>. • "cmdstan-laplace": Laplace approximation via <code>cmdstanr::laplace()</code>. Defaults to <code>character(0)</code>. Any value other than the five above (including the empty default) falls back to MCMC via <code>rstan::sampling()</code>. Ignored for JAGS models, which always use <code>rjags</code>.
niters_warmup	A count of the number of warmup iterations. The default is to use the same number of iterations as niters. This is currently only implemented for Stan models.

Details

For Stan models using a `cmdstan-*` engine, `embr` sets a fixed set of `cmdstanr` arguments from its own parameters (listed in each engine section). These cannot be overridden via `...` and are dropped with a warning if passed. Other `cmdstanr` arguments pass through `...`.

`init` is the sole exception: it can be overridden via `...`, replacing inits generated by `gen_inits()` in `model()`. Non-MCMC engines do not use `gen_inits()`.

`nthin` and `niters_warmup` are ignored by non-MCMC engines. For pathfinder, variational, and laplace, `niters` sets the number of draws from the approximated posterior.

Value

An `mb_analysis` if data is a single data frame, or an `mb_meta_analysis` if data is a named list of data frames.

cmdstan-mcmc

Calls `cmdstanr::sample()`.

`embr-controlled`: `chains`, `parallel_chains`, `iter_warmup`, `iter_sampling`, `thin`, `data`, `seed`, `init`, `show_messages`, `show_exceptions`.

Common pass-throughs: `adapt_delta`, `max_treedepth`, `step_size`, `refresh`, `output_dir`, `save_warmup`.

cmdstan-pathfinder

Calls `cmdstanr::pathfinder()`.

`embr-controlled`: `data`, `seed`, `init`, `draws` (from `niters`), `show_messages`, `show_exceptions`.

Common pass-throughs: `num_paths`, `history_size`, `max_lbfgs_iters`, `psis_resample`, `tol_rel_grad`.

cmdstan-variational

Calls `cmdstanr::variational()`.

embr-controlled: data, seed, init, draws (from niters), show_messages, show_exceptions.

Common pass-throughs: algorithm("meanfield" or "fullrank"), iter, tol_rel_obj, eval_elbo, eta.

cmdstan-optimize

Calls `cmdstanr::optimize()`.

embr-controlled: data, seed, init, show_messages, show_exceptions.

Common pass-throughs: algorithm("lbfgs", "bfgs", or "newton"), iter, tol_obj, history_size, jacobian.

cmdstan-laplace

Calls `cmdstanr::laplace()`.

embr-controlled: data, seed, init, mode, draws (from niters), show_messages, show_exceptions.

Common pass-throughs: jacobian, opt_args.

rstan

For Stan models without a stan_engine matching one of the cmdstan options above, `rstan::sampling()` is called. Pass its arguments through ...; for example, `control = list(adapt_delta = 0.95)`.

See Also

- The [analyse article](#) for a walkthrough of the fitting workflow.
- `model()` to build a model.
- `set_analysis_mode()` to set session-wide sampling defaults.
- `analyse.mb_models()` to fit a list of models simultaneously.
- `analyse.character()` to fit from raw model code.
- `predict.mb_analysis()` and `coef.mb_analysis()` to summarise an analysis.
- `reanalyse()` to refit with different sampling settings.

Examples

```
## Not run:
# Stan model with cmdstanr MCMC
analysis <- analyse(stan_model, data,
  stan_engine = "cmdstan-mcmc",
  nchains = 4, niters = 1000
)

# JAGS model
analysis <- analyse(jags_model, data, nchains = 4, niters = 2000)
```

```

# Engine-specific argument via ...
analysis <- analyse(stan_model, data,
  stan_engine = "cmdstan-mcmc",
  adapt_delta = 0.99
)

# Override initial values for cmdstanr MCMC
init_fun <- function() list(bIntercept = 0)
analysis <- analyse(stan_model, data,
  stan_engine = "cmdstan-mcmc",
  init = init_fun
)

# Multiple datasets
analyses <- analyse(model, list(dataset1 = data1, dataset2 = data2))

## End(Not run)

```

analyse.character	<i>Analyse from Character Code</i>
-------------------	------------------------------------

Description

A convenience that builds the model and fits it in one call.

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),
  seed = sample.int(.Machine$integer.max, 1),
  stan_engine = getOption("mb.stan_engine", character(0)),
  niters_warmup = niters,
  ...
)

```

Arguments

<code>x</code>	A character string of Stan or JAGS model code.
<code>data</code>	A data frame.
<code>select_data</code>	A named list specifying columns to select with their classes, values, transforms, and scaling options. Passed to <code>model()</code> .
<code>nchains</code>	A count of the number of chains (default: 3).
<code>niters</code>	A count of the number of iterations to save per chain (default: 1000).
<code>nthin</code>	A count of the thinning interval.
<code>parallel</code>	A flag indicating whether to perform the analysis in parallel if possible.
<code>quiet</code>	A flag indicating whether to disable messages and warnings, including sampling progress.
<code>glance</code>	A flag indicating whether to print a model summary.
<code>beep</code>	A flag indicating whether to beep on completion of the analysis.
<code>seed</code>	A positive whole number specifying the seed to use. The default is random. This is currently only implemented for Stan models.
<code>stan_engine</code>	A string selecting the Stan engine: • "cmdstan-mcmc": MCMC via <code>cmdstanr::sample()</code>. • "cmdstan-pathfinder": pathfinder via <code>cmdstanr::pathfinder()</code>. • "cmdstan-variational": variational ADVI via <code>cmdstanr::variational()</code>. • "cmdstan-optimize": optimization via <code>cmdstanr::optimize()</code>. • "cmdstan-laplace": Laplace approximation via <code>cmdstanr::laplace()</code>. Defaults to <code>character(0)</code>. Any value other than the five above (including the empty default) falls back to MCMC via <code>rstan::sampling()</code>. Ignored for JAGS models, which always use <code>rjags</code>.
<code>niters_warmup</code>	A count of the number of warmup iterations. The default is to use the same number of iterations as <code>niters</code> . This is currently only implemented for Stan models.
<code>...</code>	Additional arguments passed to <code>analyse()</code> .

Details

Only `select_data` is forwarded to `model()`. If you need to set `new_expr`, `random_effects`, `new_expr_vec`, or `gen_inits`, build the model with `model()` directly and pass it to `analyse()`.

Value

An `mb_analysis` or `mb_meta_analysis`. See `analyse()` for details.

See Also

`analyse()` for full argument documentation and engine details.

analyse.mb_models	<i>Analyse Multiple Models</i>
-------------------	--------------------------------

Description

Fits each model in an `mb_models` list against the same data, returning an `mb_analyses` (or `mb_meta_analyses` if data is a list of data frames).

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),
  seed = sample.int(.Machine$integer.max, 1),
  stan_engine = getOption("mb.stan_engine", character(0)),
  niters_warmup = niters,
  ...
)
```

Arguments

<code>x</code>	An <code>mb_models</code> list.
<code>data</code>	A data frame, or a named list of data frames for meta-analysis.
<code>nchains</code>	A count of the number of chains (default: 3).
<code>niters</code>	A count of the number of iterations to save per chain (default: 1000).
<code>nthin</code>	A count of the thinning interval.
<code>parallel</code>	A flag indicating whether to perform the analysis in parallel if possible.
<code>quiet</code>	A flag indicating whether to disable messages and warnings, including sampling progress.
<code>glance</code>	A flag indicating whether to print a model summary.
<code>beep</code>	A flag indicating whether to beep on completion of the analysis.
<code>seed</code>	A positive whole number specifying the seed to use. The default is random. This is currently only implemented for Stan models.
<code>stan_engine</code>	A string selecting the Stan engine: • <code>"cmdstan-mcmc"</code>: MCMC via <code>cmdstanr::sample()</code>.

- "cmdstan-pathfinder": pathfinder via `cmdstanr::pathfinder()`.
- "cmdstan-variational": variational ADVI via `cmdstanr::variational()`.
- "cmdstan-optimize": optimization via `cmdstanr::optimize()`.
- "cmdstan-laplace": Laplace approximation via `cmdstanr::laplace()`.

Defaults to `character(0)`. Any value other than the five above (including the empty default) falls back to MCMC via `rstan::sampling()`. Ignored for JAGS models, which always use `rjags`.

`niters_warmup` A count of the number of warmup iterations. The default is to use the same number of iterations as `niters`. This is currently only implemented for Stan models.

`...` Additional arguments passed to `analyse()`.

Value

An `mb_analyses` (single data frame) or `mb_meta_analyses` (list of data frames).

See Also

`analyse()` for full argument documentation and engine details.

<code>analyse_residuals</code>	<i>Analyse Residuals</i>
--------------------------------	--------------------------

Description

Analyse Residuals

Usage

```
analyse_residuals(x)
```

Arguments

`x` The `mb_analysis` object to analyse the residuals for.

analyses	<i>MB Analyses</i>
----------	--------------------

Description

Creates an object inheriting from class mb_analyses.

Usage

analyses(...)

Arguments

... Named objects.

Value

An object inheriting from class mb_analyses.

as.analyses	<i>Coerce to an mb_analyses object</i>
-------------	--

Description

Coerce to an mb_analyses object

Usage

as.analyses(x, ...)

Arguments

x object to coerce.
... Unused.

as.model	<i>Coerce to an mb_model object</i>
----------	-------------------------------------

Description

Coerce to an mb_model object

Usage

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

Arguments

x	object to coerce.
...	Unused.

as.models	<i>Coerce to an mb_models object</i>
-----------	--------------------------------------

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

<code>model</code>	An object.
<code>data</code>	Data.
<code>drops</code>	A list of character vectors specifying the scalar parameters to consider.
<code>conf_level</code>	A number specifying the confidence level. By default 0.95.
<code>beep</code>	A flag indicating whether to beep on completion of the analysis.
<code>...</code>	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	<i>Check MB Code</i>
---------------	----------------------

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	<i>Check MB Model</i>
----------------	-----------------------

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	<i>Check Model Parameters</i>
------------------	-------------------------------

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	<i>Check Uniquely Named List</i>
---------------------------	----------------------------------

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 coefficient terms with the model averaged estimate, the Akaike's weight and the proportion of models including the term.

coef.mb_analysis	<i>Coef JAGS Analysis</i>
------------------	---------------------------

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 = TRUE,
  directional_information = 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	Must be TRUE. simplify = FALSE is defunct as of embr 1.1.0.

directional_information	A flag specifying whether the svalue column for a Bayesian analysis should be calculated using <code>extras::directional_information()</code> instead of <code>extras::svalue()</code> . The default value will change from FALSE to TRUE in a future release; set the argument explicitly to avoid the deprecation warning. Ignored for frequentist analyses where the svalue is always $-\log_2(\text{pvalue})$.
...	Not used.

Value

A tidy tibble of the coefficient terms with the columns indicating the term, estimate, lower and upper confidence or credible intervals and svalue.

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 coefficient 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

<code>object</code>	The <code>mb_analysis</code> object.
<code>param_type</code>	A flag specifying whether 'fixed', 'random' or 'derived' terms.
<code>include_constant</code>	A flag specifying whether to include constant terms.
<code>conf_level</code>	A number specifying the confidence level. By default 0.95.
<code>estimate</code>	The function to use to calculating the estimate for Bayesian models.
<code>parallel</code>	A flag indicating whether to using parallel backend provided by foreach.
<code>beep</code>	A flag indicating whether to beep on completion of the analysis.
<code>simplify</code>	A flag specifying whether to simplify with <code>svalue</code> .
<code>...</code>	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

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

Value

A tidy tibble.

comment_string	<i>Comment String</i>
----------------	-----------------------

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	<i>Data Set</i>
----------	-----------------

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	<i>Density99</i>
-----------	------------------

Description

dummy data

Usage

density99

Format

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

drop_pars	<i>Drop pars</i>
-----------	------------------

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	<i>Elapsed</i>
---------	----------------

Description

Get elapsed duration.

Usage

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

Arguments

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

estimates.mb_analysis	<i>Estimates</i>
-----------------------	------------------

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	<i>Fitted Values</i>
--------------------	----------------------

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	<i>Get Analysis Mode</i>
-------------------	--------------------------

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 thinning 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.prop_divergent A number specifying the maximum proportion of divergent transitions.

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	<i>Retrieve model</i>
-----------	-----------------------

Description

Constructs a new analysis object

Usage

```
get_model(analysis)
```

Arguments

analysis An object of class "mb_analysis".

IC	<i>Information Criterion</i>
----	------------------------------

Description

Calculates Information Criterion for an analysis.

Usage

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

Arguments

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

Value

The Information Criterion as a number.

is.lmb_analysis	<i>Is a LMB Analysis</i>
-----------------	--------------------------

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	<i>Is MB Analyses?</i>
----------------	------------------------

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	<i>Is MB Analysis?</i>
----------------	------------------------

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	<i>Is syntactic</i>
--------------	---------------------

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	<i>Test if is bayesian</i>
-------------	----------------------------

Description

Test if is bayesian

Usage

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

Arguments

x the object.
... Unused

Value

A flag indicating wether bayesian.

is_frequentist	<i>Test if is frequentist</i>
----------------	-------------------------------

Description

Test if is frequentist

Usage

is_frequentist(x)

Arguments

x the object.

Value

A flag indicating whether frequentist

is_namedlist	<i>Is Named List</i>
--------------	----------------------

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	<i>Load Model</i>
------------	-------------------

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.

log_lik_draws.mb_analysis	<i>Extract log likelihood draws</i>
---------------------------	-------------------------------------

Description

Extract log likelihood from fitted model and return as a draws object. Adapted from the priorsense package.

Usage

```
## S3 method for class 'mb_analysis'
log_lik_draws(x, joint = FALSE, log_lik_name = "log_lik", ...)
```

Arguments

x	The mb_analysis object.
joint	A flag indicating whether to return the joint log likelihood or array, default is FALSE.
log_lik_name	A string of the name of the parameter corresponding to the log likelihood, default is "log_lik".
...	Unused.

Value

A draws_array object containing log_lik values.

```
log_prior_draws.mb_analysis
```

Extract log prior draws

Description

Extract log likelihood from fitted model and return as a draws object. Adapted from the priorsense package.

Usage

```
## S3 method for class 'mb_analysis'
log_prior_draws(x, joint = FALSE, log_prior_name = "lprior", ...)
```

Arguments

x	The mb_analysis object.
joint	A flag indicating whether to return the joint log likelihood or array, default is FALSE.
log_prior_name	A string of the name of the parameter corresponding to the log prior, default is "lprior".
...	Unused.

Value

A draws_array object containing log_prior values.

```
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_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),
  ...
)

## 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),
  ...
)
```

Arguments

<code>object</code>	An object inheriting from class <code>mb_analysis</code> .
<code>new_data</code>	A data frame at which to derive the term. Pass character(0) to extract a scalar term from <code>new_expr</code> .

<code>new_expr</code>	An R expression (e.g. <code>{ ... }</code>) or a character string of R code specifying the predictive relationship. If <code>NULL</code> , uses the expression set in <code>model()</code> and stored in the <code>mb_analysis</code> object.
<code>new_values</code>	A named list of new or replacement values to pass to <code>new_expr</code> .
<code>term</code>	A string of the term in <code>new_expr</code> .
<code>modify_new_data</code>	A single argument function to modify new data (in list form) immediately prior to calculating <code>new_expr</code> .
<code>ref_data</code>	A flag or a data frame with 1 row indicating the reference values for calculating the effects size. If <code>FALSE</code> , no reference applied. If <code>TRUE</code> , the reference is a 1 row data.frame with all variables held at reference value.
<code>ref_fun2</code>	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
<code>random_effects</code>	A named list specifying the random effects and the associated factors.
<code>new_expr_vec</code>	A flag specifying whether to vectorize the <code>new_expr</code> code.
<code>parallel</code>	A flag indicating whether to do predictions using parallel backend provided by <code>foreach</code> .
<code>quiet</code>	A flag indicating whether to disable tracing information.
<code>...</code>	Additional arguments.

Value

A object of class `mcmcr`.

See Also

- The [prediction article](#) for worked patterns including arithmetic on `mcmcr` posteriors.
- `predict.mb_analysis()` for tidy posterior summaries at new covariate values.
- `mcmc_derive_data.mb_analysis()` for raw MCMC samples paired with `new_data` and group-level summaries.
- `mcmcr::combine_samples()` for combining MCMC samples across independent analyses on shared data keys.

Examples

```
## Not run:
# Pull multiple scalar terms in one call by regex. Quantities defined in
# new_expr are usually easiest to extract individually with predict();
# mcmc_derive() is most useful when (a) you want raw mcmcr samples for
# downstream operations predict() does not support (probability
# statements, custom quantiles) or (b) you want to compose multiple terms
# with arbitrary arithmetic. See vignette("prediction") for worked
# patterns.

scalars <- mcmc_derive(
  analysis,
```

```

    new_data = character(0),
    term = "^(eBaseCount|eRestoredEffect)$"
  )
  coef(scalars)

# Custom expression with injected scalar constants
as.mcmcr(analysis) |>
  mcmc_derive(
    expr = "biomass <- exp(bIntercept) * mean_mass_g",
    values = list(mean_mass_g = 250)
  ) |>
  coef()

## End(Not run)

```

```

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

<code>object</code>	An object inheriting from class <code>mb_analysis</code> .
<code>new_data</code>	The data frame to calculate the predictions for.
<code>new_expr</code>	An R expression (e.g. <code>{ ... }</code>) or a character string of R code specifying the predictive relationship.

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

Value

A object of class `mcmc_data`.

See Also

[mcmc_derive_data.mb_analysis\(\)](#) for full argument documentation and examples.

`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

<code>object</code>	An object inheriting from class <code>mb_analysis</code> .
<code>new_data</code>	A data frame at which to derive the term. Pass <code>character(0)</code> to extract a scalar term from <code>new_expr</code> .
<code>new_expr</code>	An R expression (e.g. <code>{ ... }</code>) or a character string of R code specifying the predictive relationship. If <code>NULL</code> , uses the expression set in <code>model()</code> and stored in the <code>mb_analysis</code> object.
<code>new_values</code>	A named list of new or replacement values to pass to <code>new_expr</code> .
<code>term</code>	A string of the term in <code>new_expr</code> .
<code>modify_new_data</code>	A single argument function to modify new data (in list form) immediately prior to calculating <code>new_expr</code> .
<code>ref_data</code>	A flag or a data frame with 1 row indicating the reference values for calculating the effects size. If <code>FALSE</code> , no reference applied. If <code>TRUE</code> , the reference is a 1 row data.frame with all variables held at reference value.
<code>ref_fun2</code>	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
<code>new_expr_vec</code>	A flag specifying whether to vectorize the <code>new_expr</code> code.
<code>random_effects</code>	A named list specifying the random effects and the associated factors.
<code>parallel</code>	A flag indicating whether to do predictions using parallel backend provided by <code>foreach</code> .
<code>quiet</code>	A flag indicating whether to disable tracing information.
<code>beep</code>	A flag indicating whether to beep on completion of the analysis.
<code>...</code>	Additional arguments.

Value

A object of class `mcmc_data`.

See Also

- The [prediction article](#) for worked patterns including group-level summaries.
- `predict.mb_analysis()` for tidy posterior summaries at new covariate values.
- `mcmc_derive.mb_analysis()` for scalar derived quantities and arithmetic on `mcmc` posteriors.
- `mcmc::combine_samples()` for combining MCMC samples across independent analyses on shared data keys.
- `newdata::xnew_data()` for building covariate grids.

Examples

```
## Not run:
# `analysis` is a fitted mb_analysis with factors site, treatment and
# new_expr that defines per-row term `eCount`.
data <- data_set(analysis)

# Per-group posterior summaries via group_by() + summarise().
# Default .fun = sum; use mean for the per-treatment posterior mean.
mcmc_derive_data(analysis, new_data = data, term = "^eCount$") |>
  group_by(treatment) |>
  summarise(.fun = mean) |>
  coef()

# Custom summarise function, e.g. range within group
mcmc_derive_data(analysis, new_data = data, term = "^eCount$") |>
  group_by(treatment) |>
  summarise(.fun = function(x) max(x) - min(x)) |>
  coef()

## End(Not run)
```

model

Define a Model

Description

Define a model, including code, data, monitoring, and post-fitting specification. The returned `mb_model` is consumed by [analyse\(\)](#) and downstream by [predict.mb_analysis\(\)](#) and [mcmc_derive_data\(\)](#).

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

<code>x</code>	An <code>mb_code</code> object built with <code>mb_code()</code> , or <code>NULL</code> (the default). When <code>NULL</code> , the model code is built inline from <code>code</code> . Passing a character string or an <code>mb_analysis</code> is deprecated.
<code>...</code>	These dots are for future extensions and must be empty.
<code>code</code>	Model code (Stan or JAGS expression or string) to pass to <code>mb_code()</code> . Only used when <code>x = NULL</code> .
<code>gen_inits</code>	A single-argument function returning a named list of initial values per chain. Receives the modified data list (i.e. after <code>select_data</code> , rescaling, type coercion, factor-count and <code>nObs</code> injection, and <code>modify_data</code>). Returning <code>list()</code> lets the backend fall back to its own defaults (JAGS samples from priors; Stan uses its random init). Non-MCMC <code>cmdstan-*</code> engines ignore it.
<code>random_effects</code>	Named list mapping parameter names in the model code to one or more grouping factor columns from <code>select_data</code> . Each named factor must appear in <code>select_data</code> and must not also be in <code>derived</code> .
<code>fixed</code>	A string of a regular expression of parameter names to monitor as fixed effects.
<code>derived</code>	A character vector of derived parameters to monitor.
<code>select_data</code>	A named list specifying the columns to select and their expected classes and values as well as transformations and scaling options. See the select_data section of <code>model()</code> .
<code>center</code>	A character vector of the columns to center.
<code>scale</code>	A character vector of the columns to scale (after centering).
<code>modify_data</code>	A single argument function to modify the data (in list form) immediately prior to the analysis.
<code>nthin</code>	An integer specifying the thinning interval.
<code>new_expr</code>	An R expression or character string of R code defining the predictive relationships and derived quantities. See the new_expr section of <code>model()</code> .
<code>new_expr_vec</code>	Flag controlling whether to vectorise the <code>new_expr</code> code via <code>mcmcderive::expression_vectorize()</code> . See the new_expr section of <code>model()</code> for safe patterns and silent fallbacks.
<code>modify_new_data</code>	SA single argument function to modify new data (in list form) immediately prior to calculating <code>new_expr</code> .
<code>drops</code>	A list of character vectors naming scalar parameters to fix at 0 in the model.

Details

The data passed to `analyse()` is transformed through a fixed pipeline before reaching the model code: `select_data` selects columns and applies any rescaling suffixes; logical, date, and `difftime` columns are coerced to numeric; one `n<Factor>` count is injected per factor; `nObs` is injected; and finally the user's `modify_data` function runs against the resulting list. `gen_inits` is then invoked against this same modified list, once per chain, to seed initial values.

The monitored parameter set is the union of: names matching the `fixed` regex, the names of `random_effects`, and `derived`. The default `fixed = "^[^e]"` matches everything not starting with `e`. Names listed in `random_effects` are always monitored regardless of the regex.

Value

An object inheriting from class "mb_model".

select_data

Named list mapping data column names to type or range specifications: 1L for integer, 1 for numeric, factor("") for factor, TRUE for logical. Where an explicit check is wanted, use a range form such as c(0L, 100L) and append NA to allow missing values: c(0L, 100L, NA).

Appending a suffix to a column name requests a rescaling transformation; the original column is replaced with the transformed version in the analysis dataset.

Suffix	Transformation
(none)	Raw, untransformed
+	Subtract the mean (center)
-	Subtract the minimum (shift to 0)
=	Subtract the minimum and add 1 (shift to 1)
/	Divide by SD (scale)
*	Subtract the mean and divide by SD (standardise)

* is conventional for continuous covariates and + for year or date variables. Never apply a suffix to the response. The center and scale character-vector arguments are kept for backwards compatibility and emit a deprecation warning; prefer select_data suffixes.

new_expr

An R expression (e.g. { ... }) or a character string of R code that is evaluated post-fitting against the MCMC draws. It has access to the sampled parameters, the modified data list, and extras helpers such as log_lik_* and res_*. The expression typically defines the named terms used by downstream tooling:

- prediction[i] for `predict.mb_analysis()`
- fit[i] for `fitted.mb_analysis()` and for residuals
- log_lik[i] for information criteria and prior sensitivity
- lprior[k] for prior sensitivity

The scale of fit[i] is model-specific: define it to match the signature of the res_* and log_lik_* helpers you use (e.g. response-scale for res_neg_binom, log-scale for res_student on log-normal data). Scalar derived quantities defined outside the loop are extractable via `predict(analysis, new_data = character(0), term = "myScalar")`.

If new_expr is supplied as a string without wrapping braces they are added automatically. Setting new_expr_vec = TRUE vectorises the for-loop body via `mcmcderive::expression_vectorize()` for substantial speed gains. Vectorisation silently falls back to the un-vectorised loop (no error) when the loop body contains `sum()`, nested for loops, or dynamic range indexing such as `eK[year[i]:(year[i] + n)]`. Safe patterns: linear predictors built from indexed terms, single-level (`bSite[site[i]]`) and multi-level (`bParam[site[i], annual[i]]`) indexing, link-function assignments (`log(eY[i]) <- ...`), and polynomial terms on standardised predictors.

See Also

- The [analyse article](#) for a walkthrough including `select_data` transformations and `new_expr` structure.
- `analyse()` to fit the model to data.
- `analyse.character()` for a shortcut that builds and fits in one call.
- `update_model()` to change arguments on an existing `mb_model`.

Examples

```
## Not run:
# Stan model with a standardised continuous covariate and a site
# random effect.
count_model <- model(
  code = count_code, # a string or read_file("stan/count.stan")
  select_data = list(
    count = 1L,
    site = factor("a"),
    `temperature*` = 1
  ),
  random_effects = list(z_bSite = "site"),
  new_expr = {
    bSite <- z_bSite * sSite
    for (i in 1:nObs) {
      log(eCount[i]) <- bIntercept + bTemp * temperature[i] +
        bSite[site[i]]
      prediction[i] <- eCount[i]
      fit[i] <- eCount[i]
      log_lik[i] <- log_lik_neg_binom(count[i], eCount[i], bPhi)
    }
  },
  new_expr_vec = TRUE
)

# Reshape multi-pass data into a matrix and add a per-row pass count
# before the analysis runs; seed eAbundance with the observed totals.
depletion_model <- model(
  code = depletion_code,
  select_data = list(
    Pass1 = 1L, Pass2 = 1L, Pass3 = 1L,
    site = factor("a")
  ),
  modify_data = function(data) {
    Pass <- as.matrix(data[c("Pass1", "Pass2", "Pass3")])
    data$Pass <- Pass
    data$nPass <- ncol(Pass)
    data$Pass1 <- data$Pass2 <- data$Pass3 <- NULL
    data
  },
  gen_inits = function(data) {
    list(eAbundance = rowSums(data$Pass, na.rm = TRUE) + 1)
  }
)
```

```
)  
  
## End(Not run)
```

models	<i>MB Models</i>
--------	------------------

Description

Creates an object inheriting from class mb_models.

Usage

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

Arguments

... Named objects.

Value

An object inheriting from class mb_models.

modify_data	<i>Modify Data</i>
-------------	--------------------

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	<i>Modify New Data</i>
-----------------	------------------------

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	<i>pars to monitor</i>
---------	------------------------

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	<i>MB Analysis</i>
--------------	--------------------

Description

Constucts a new analysis object

Usage

new_analysis(x, class)

Arguments

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

new_expr	<i>new_expr</i>
----------	-----------------

Description

new_expr

Usage

new_expr(object, ...)

Arguments

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

Value

The new_expr

new_expr<-	<i>new_expr set</i>
------------	---------------------

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	<i>Total Number of MCMC simulations generated (including warmup)</i>
-------	--

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.
nchains	A count of the number of chains (default: 3).
niters	A count of the number of iterations to save per chain (default: 1000).
nthin	A count of the thinning interval.
parallel	A flag indicating whether to perform the analysis in parallel if possible.
quiet	A flag indicating whether to disable messages and warnings, including sampling progress.
glance	A flag indicating whether to print a model summary.
beep	A flag indicating whether to beep on completion of the analysis.
seed	A positive whole number specifying the seed to use. The default is random. This is currently only implemented for Stan models.

niters_warmup	A count of the number of warmup iterations. The default is to use the same number of iterations as niters. This is currently only implemented for Stan models.
stan_engine	A string selecting the Stan engine: • "cmdstan-mcmc": MCMC via <code>cmdstanr::sample()</code>. • "cmdstan-pathfinder": pathfinder via <code>cmdstanr::pathfinder()</code>. • "cmdstan-variational": variational ADVI via <code>cmdstanr::variational()</code>. • "cmdstan-optimize": optimization via <code>cmdstanr::optimize()</code>. • "cmdstan-laplace": Laplace approximation via <code>cmdstanr::laplace()</code>. Defaults to <code>character(0)</code> . Any value other than the five above (including the empty default) falls back to MCMC via <code>rstan::sampling()</code> . Ignored for JAGS models, which always use <code>rjags</code> .
...	Unused.

plot.mb_analysis	<i>Plot Analysis</i>
------------------	----------------------

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	<i>Plot Data</i>
-----------	------------------

Description

Plot Data

Usage

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

Arguments

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

plot_residuals	<i>Plot Residuals</i>
----------------	-----------------------

Description

Plot Residuals

Usage

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

Arguments

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

posterior_predictive_check	<i>Posterior Predictive Check</i>
----------------------------	-----------------------------------

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.

```
powerscale_sensitivity.mb_analysis
```

Run Power-Scaling Sensitivity Analysis

Description

Calculates the prior/likelihood sensitivity based on power-scaling perturbations. This is done using importance sampling (and optionally moment matching).

Usage

```
## S3 method for class 'mb_analysis'
powerscale_sensitivity(x, ...)
```

Arguments

x	The mb_analysis object.
...	Arguments passed to methods to priorsense::powerscale_sensitivity() . Alternative functions for log_lik_fn and log_prior_fn cannot be used.

Value

Table of sensitivity values for each specified variable.

predict.mb_analyses *Predict from a mb_analyses object*

Description

Returns IC-weighted model-averaged predictions across an mb_analyses list.

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),
  ...
)
```

Arguments

object	An object inheriting from class mb_analysis.
new_data	A data frame at which to derive the term. Pass character(0) to extract a scalar term from new_expr.
new_expr	An R expression (e.g. { ... }) or a character string of R code specifying the predictive relationship. If NULL, uses the expression set in <code>model()</code> and stored in the mb_analysis object.
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. If FALSE, no reference applied. If TRUE, the reference is a 1 row data.frame with all variables held at reference value.
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.
...	Additional arguments.

Value

A data frame with one row per row of new_data.

See Also

[predict.mb_analysis\(\)](#) for full argument documentation and examples.

predict.mb_analysis *Predict from a mb_analysis object*

Description

Summarises the posterior predictive distribution at new covariate values. Returns a tidy data frame with point estimates and compatibility limits.

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),
  ...
)
```

Arguments

<code>object</code>	An object inheriting from class <code>mb_analysis</code> .
<code>new_data</code>	A data frame at which to derive the term. Pass <code>character(0)</code> to extract a scalar term from <code>new_expr</code> .
<code>new_expr</code>	An R expression (e.g. <code>{ ... }</code>) or a character string of R code specifying the predictive relationship. If <code>NULL</code> , uses the expression set in <code>model()</code> and stored in the <code>mb_analysis</code> object.
<code>new_values</code>	A named list of new or replacement values to pass to <code>new_expr</code> .
<code>term</code>	A string of the term in <code>new_expr</code> .
<code>conf_level</code>	A number specifying the confidence level. By default 0.95.
<code>modify_new_data</code>	A single argument function to modify new data (in list form) immediately prior to calculating <code>new_expr</code> .
<code>ref_data</code>	A flag or a data frame with 1 row indicating the reference values for calculating the effects size. If <code>FALSE</code> , no reference applied. If <code>TRUE</code> , the reference is a 1 row data.frame with all variables held at reference value.
<code>ref_fun2</code>	A function whose first argument takes a vector of two numbers and returns a scalar of a metric of the difference between them.
<code>new_expr_vec</code>	A flag specifying whether to vectorize the <code>new_expr</code> code.
<code>random_effects</code>	A named list specifying the random effects and the associated factors.
<code>parallel</code>	A flag indicating whether to do predictions using parallel backend provided by <code>foreach</code> .
<code>quiet</code>	A flag indicating whether to disable tracing information.
<code>...</code>	Additional arguments.

Details

`new_data` defaults to `data_set(object)` (the analysis dataset). Build covariate grids with `newdata::xnew_data()` and related helpers (`newdata::xnew_seq()`, `newdata::xcast()`, `newdata::xobs_only()`) for explicit control over which covariates vary. A character vector of column names is also accepted as a shortcut: `predict(analysis, "year")` is equivalent to `predict(analysis, new_data = newdata::xnew_data(data_set(year))`, which generates a grid varying year and holding the rest at their reference values. Covariates not specified are held at a reference value: mean for continuous, first level for factors.

`term` selects the quantity defined in the model's `new_expr` to calculate (default "prediction"). Pass `new_data = character(0)` to extract a scalar quantity.

Value

A data frame with one row per row of `new_data`, containing the posterior summary columns produced by `coef.mcmc_data()` and all columns of `new_data`.

See Also

- The [prediction article](#) for detailed examples of prediction patterns.
- `predict.mb_analyses()` for predictions on meta-analysis object.
- `mcmc_derive_data.mb_analysis()` for raw MCMC samples paired with `new_data`, regex term matching, and group-level summaries via `group_by()` + `summarise()`.
- `mcmc_derive.mb_analysis()` for scalar derived quantities and arithmetic on mcmc posteriors.
- `mcmcr::combine_samples()` for combining MCMC samples across independent analyses on shared data keys.
- `newdata::xnew_data()` for building covariate grids.

Examples

```
## Not run:
library(newdata)
data <- data_set(analysis)

# Character shortcut: vary a single column, hold others at reference
predict(analysis, "temperature")

# Equivalent explicit form
xnew_data(data, temperature) |>
  predict(analysis, new_data = _)

# Set custom sequence for continuous covariate
xnew_data(data, xnew_seq(temperature, length_out = 5)) |>
  predict(analysis, new_data = _)

# Predict for all factor levels and set specific continuous reference value
xnew_data(data, site, temperature = 5) |>
  predict(analysis, new_data = _)

# Predict only for observed combinations of factor levels
xnew_data(data, xobs_only(site, annual)) |>
  predict(analysis, new_data = _)

# Proportional change relative to a 1-row reference state
ref <- xnew_data(data, xcast(treatment = "control"))
xnew_data(data, treatment) |>
  predict(analysis, new_data = _, ref_data = ref)

# Predict for first level of random effect levels rather than 'typical'
xnew_data(data, temperature) |>
  predict(analysis, new_data = _, random_effects = FALSE)

# Extract a scalar derived quantity from new_expr
predict(analysis, new_data = character(0), term = "eBaseCount")

## End(Not run)
```

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),
  ...
)
```

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.
...	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	<i>Random Effects</i>
----------------	-----------------------

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	<i>Reanalyse</i>
-----------	------------------

Description

Reanalyse an analysis.

Usage

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

Arguments

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

reanalyse.mb_analyses	<i>Reanalyse</i>
-----------------------	------------------

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),
  seed = sample.int(.Machine$integer.max, 1),
  ...
)
```

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.
seed	A positive whole number specifying the seed to use. The default is random. This is currently only implemented for Stan models.
...	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

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

`reanalyse1`*Reanalyse*

Description

Reanalyse an analysis. For user to override.

Usage

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

Arguments

<code>object</code>	The object to reanalyse.
<code>parallel</code>	A flag indicating whether to perform the reanalysis in parallel.
<code>quiet</code>	A flag indicating whether to capture output.
<code>...</code>	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

<code>object</code>	The MB analysis object.
<code>type</code>	A string of the residual type.
<code>...</code>	Unused.

Details

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

Value

The analysis data set with the residual values.

rm_comments	<i>Removes comments from the template strings(s) of an mb object or a character vector.</i>
-------------	---

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	<i>Sample Size</i>
-------------	--------------------

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	<i>Scalar Named List</i>
--------------	--------------------------

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	<i>Multiply Standard Deviation of Priors By</i>
--------------	---

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 messages and warnings, including sampling progress.
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	<i>Select and Rescale Data</i>
---------------------	--------------------------------

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 <code>mb_model</code> .
data2	The base data.

Value

The modified data in list form.

sensitivity	<i>Summarize model sensitivity</i>
-------------	------------------------------------

Description

Summarize model sensitivity

Usage

```
sensitivity(  
  x,  
  by = "term",  
  param_type = "all",  
  mb.prior_cjs = getOption("mb.prior_cjs", 0.1),  
  mb.lik_cjs = getOption("mb.lik_cjs", 0.05),  
  ...  
)
```

Arguments

x	The mb_analysis object.
by	A string indicating whether to determine by "term", "parameter", or "all".
param_type	A string specifying which parameters to include: 'fixed', 'random', 'derived', 'primary', or 'all'.
mb.prior_cjs	A number specifying the CJS threshold for weak prior classification.
mb.lik_cjs	A number specifying the CJS threshold for strong data classification.
...	Arguments passed to add_sensitivity() .

Value

A tibble summarizing the sensitivity of the analysis object.

set_analysis_mode	<i>Set Analysis Mode</i>
-------------------	--------------------------

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 thinning 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.prop_divergent** A number specifying the maximum proportion of divergent transitions.
- 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	<i>Simulate Residuals</i>
--------------------	---------------------------

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	<i>Sort Analyses by IC</i>
------------	----------------------------

Description

Sort Analyses by IC

Usage

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

Arguments

x	the object.
...	Unused

Value

The sorted object

sort_nlist	<i>Sort Named List</i>
------------	------------------------

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	<i>Template</i>
----------	-----------------

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<-	<i>Set Template</i>
------------	---------------------

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	<i>Terms</i>
-------------------	--------------

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	<i>Update MB Model</i>
--------------	------------------------

Description

Updates an object inheriting 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 = NULL,
  modify_new_data = NULL,
  drops = NULL,
  ...
)
```

Arguments

<code>model</code>	The model to update.
<code>code</code>	A string of the model template or an object inheriting from class <code>mb_code</code> .
<code>gen_inits</code>	A single-argument function returning a named list of initial values per chain. Receives the modified data list (i.e. after <code>select_data</code> , rescaling, type coercion, factor-count and <code>nObs</code> injection, and <code>modify_data</code>). Returning <code>list()</code> lets the backend fall back to its own defaults (JAGS samples from priors; Stan uses its random init). Non-MCMC <code>cmdstan-*</code> engines ignore it.
<code>random_effects</code>	Named list mapping parameter names in the model code to one or more grouping factor columns from <code>select_data</code> . Each named factor must appear in <code>select_data</code> and must not also be in <code>derived</code> .
<code>fixed</code>	A string of a regular expression of parameter names to monitor as fixed effects.
<code>derived</code>	A character vector of derived parameters to monitor.
<code>select_data</code>	A named list specifying the columns to select and their expected classes and values as well as transformations and scaling options. See the select_data section of model() .

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	An integer specifying the thinning interval.
new_expr	An R expression or character string of R code defining the predictive relationships and derived quantities. See the new_expr section of model() .
new_expr_vec	Flag controlling whether to vectorise the new_expr code via mcmcderive::expression_vectorize() . See the new_expr section of model() for safe patterns and silent fallbacks.
modify_new_data	SA single argument function to modify new data (in list form) immediately prior to calculating new_expr.
drops	A list of character vectors naming scalar parameters to fix at 0 in the model.
...	These dots are for future extensions and must be empty.

Value

An object inheriting from class `mb_model`.

Index

* datasets

- density99, 28
- add_analyses, 4
- add_models, 5
- add_sensitivity, 5
- add_sensitivity(), 75
- analyse, 6
- analyse(), 10, 12, 48, 49, 51
- analyse.character, 9
- analyse.character(), 8, 51
- analyse.mb_models, 11
- analyse.mb_models(), 8
- analyse_residuals, 12
- analyses, 13
- as.analyses, 13
- as.model, 14
- as.models, 14
- backwards, 15
- base_model, 16
- bind_chains, 29
- bind_iterations, 29
- check_mb_analysis, 16
- check_mb_code, 17
- check_mb_model, 17
- check_model_pars, 18
- check_uniquely_named_list, 18
- cmdstanr::laplace(), 7, 8, 10, 12, 58
- cmdstanr::optimize(), 7, 8, 10, 12, 58
- cmdstanr::pathfinder(), 7, 10, 12, 58
- cmdstanr::sample(), 7, 10, 11, 58
- cmdstanr::variational(), 7, 8, 10, 12, 58
- code, 19
- coef.mb_analyses, 19
- coef.mb_analysis, 20
- coef.mb_analysis(), 8
- coef.mb_meta_analyses, 21
- coef.mb_meta_analysis, 22
- coef.mcmc_data(), 63
- coef_profile, 22
- coef_profile.mb_analyses, 23
- coef_profile.mb_analysis, 24
- coef_profile.mb_meta_analyses, 25
- coef_profile.mb_meta_analysis, 26
- collapse_chains, 29
- comment_string, 27
- data_set, 27
- density99, 28
- drop_pars, 28
- elapsed, 29
- estimates.mb_analysis, 29
- extras::directional_information(), 21
- extras::svalue(), 21
- fitted.mb_analysis, 30
- fitted.mb_analysis(), 50
- get_analysis_mode, 30
- get_model, 31
- IC, 32
- is.lmb_analysis, 32
- is.lmb_code, 33
- is.lmb_model, 33
- is.mb_analyses, 34
- is.mb_analysis, 34
- is.mb_code, 35
- is.mb_model, 35
- is.mb_models, 36
- is.mb_null_analysis, 36
- is.syntactic, 37
- is_bayesian, 37
- is_frequentist, 38
- is_namedlist, 38
- load_model, 39
- log_lik_draws.mb_analysis, 39

log_prior_draws.mb_analysis, 40
 logLik.mb_analysis, 40
 logLik.mb_null_analysis, 41

 make_all_models, 41
 mb_code, 42
 mb_code(), 49
 mcmc_derive.mb_analyses
 (mcmc_derive.mb_analysis), 43
 mcmc_derive.mb_analysis, 43
 mcmc_derive.mb_analysis(), 47, 64
 mcmc_derive_data(), 48
 mcmc_derive_data.mb_analyses, 45
 mcmc_derive_data.mb_analysis, 46
 mcmc_derive_data.mb_analysis(), 44, 46, 64
 mcmcderive::expression_vectorize(), 49, 50, 81
 mcmcr::combine_samples(), 44, 47, 64
 model, 48
 model(), 7, 8, 10, 44, 47, 49, 61, 63, 80, 81
 models, 52
 modify_data, 52
 modify_new_data, 53
 monitor, 53

 new_analysis, 54
 new_expr, 54
 new_expr<-, 55
 new_mb_code (mb_code), 42
 newdata::xcast(), 63
 newdata::xnew_data(), 47, 63, 64
 newdata::xnew_seq(), 63
 newdata::xobs_only(), 63
 ngens, 55
 nmodels, 56
 nterms.mb_analysis, 56
 nthin, 57

 params, 57
 plot.mb_analysis, 58
 plot_data, 58
 plot_residuals, 59
 posterior_predictive_check, 59
 posterior_predictive_check.mb_analysis, 60
 powerscale_sensitivity(), 5
 powerscale_sensitivity.mb_analysis, 60
 predict.mb_analyses, 61
 predict.mb_analyses(), 64
 predict.mb_analysis, 62
 predict.mb_analysis(), 8, 44, 47, 48, 50, 62
 priorsense::powerscale_sensitivity(), 60

 R2, 65
 R2.mb_analysis, 65
 random_effects, 66
 reanalyse, 67
 reanalyse(), 8
 reanalyse.mb_analyses, 67
 reanalyse.mb_analysis, 68
 reanalyse.mb_meta_analyses, 69
 reanalyse.mb_meta_analysis, 70
 reanalyse1, 71
 residuals.mb_analysis, 71
 rjags, 7, 10, 12, 58
 rm_comments, 72
 rstan::sampling(), 7, 8, 10, 12, 58

 sample_size, 72
 scalar_nlist, 73
 sd_priors_by, 73
 select_rescale_data, 74
 sensitivity, 75
 set_analysis_mode, 75
 set_analysis_mode(), 8
 simulate_residuals, 77
 sort_by_ic, 77
 sort_nlist, 78
 split_chains, 29

 template, 78
 template<-, 79
 terms.mb_analysis, 79

 update_model, 80
 update_model(), 51