

# Package ‘BayesRTMB’

September 10, 2026

**Type** Package

**Title** Bayesian Inference Using 'RTMB'

**Version** 0.4.0

**Description** Provides tools for Markov chain Monte Carlo (MCMC) and Maximum A Posteriori (MAP) estimation utilizing the 'RTMB' package. It supports various statistical models including generalized linear mixed models, factor analysis, item response theory, and multidimensional unfolding. The package allows users to easily transition between frequentist and Bayesian paradigms using a unified interface. Automatic differentiation and Laplace approximation follow Kristensen et al. (2016) <[doi:10.18637/jss.v070.i05](https://doi.org/10.18637/jss.v070.i05)>, and MCMC sampling uses the No-U-Turn Sampler described by Hoffman and Gelman (2014) <<https://jmlr.org/papers/v15/hoffman14a.html>>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**Depends** R (>= 4.1.0), RTMB

**Imports** R6, MASS

**URL** <https://github.com/norimune/BayesRTMB>,  
<https://norimune.github.io/BayesRTMB/>

**BugReports** <https://github.com/norimune/BayesRTMB/issues>

**Suggests** knitr, rmarkdown, GPArotation, future

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Hiroshi Shimizu [aut, cre]

**Maintainer** Hiroshi Shimizu <[simizu706@gmail.com](mailto:simizu706@gmail.com)>

**Repository** CRAN

**Date/Publication** 2026-09-10 12:00:02 UTC

## Contents

|                                        |    |
|----------------------------------------|----|
| ADVI_method . . . . .                  | 4  |
| bayes_factor . . . . .                 | 5  |
| beverage . . . . .                     | 6  |
| BigFive . . . . .                      | 7  |
| center_grand_mean . . . . .            | 8  |
| center_within_cluster . . . . .        | 9  |
| Classic_Fit . . . . .                  | 9  |
| conditional_effects . . . . .          | 17 |
| conditional_effects.mcmc_fit . . . . . | 18 |
| debate . . . . .                       | 19 |
| diffusion_lpdf . . . . .               | 19 |
| Dim . . . . .                          | 20 |
| distance . . . . .                     | 21 |
| distributions . . . . .                | 21 |
| ess_basic . . . . .                    | 23 |
| ess_bulk . . . . .                     | 23 |
| ess_tail95 . . . . .                   | 24 |
| exp_mod_normal_lpdf . . . . .          | 24 |
| fabs . . . . .                         | 25 |
| gaussian_process_lpdf . . . . .        | 25 |
| generate_random_init . . . . .         | 26 |
| inv_logit . . . . .                    | 27 |
| item_curve . . . . .                   | 27 |
| item_curve.RTMB_Fit_Base . . . . .     | 28 |
| item_info . . . . .                    | 28 |
| item_info.RTMB_Fit_Base . . . . .      | 29 |
| log1m . . . . .                        | 29 |
| log1m_exp . . . . .                    | 30 |
| log1p_exp . . . . .                    | 30 |
| logit . . . . .                        | 31 |
| log_det_chol . . . . .                 | 31 |
| log_mix . . . . .                      | 32 |
| log_softmax . . . . .                  | 32 |
| log_sum_exp . . . . .                  | 33 |
| log_sum_exp_matrix . . . . .           | 33 |
| lsmeans . . . . .                      | 34 |
| make_bw_from_ydif . . . . .            | 34 |
| make_glmer_re_terms . . . . .          | 35 |
| make_glmer_Z_matrix . . . . .          | 36 |
| make_init_mdu . . . . .                | 36 |
| make_ydif_from_bw . . . . .            | 37 |
| map_est . . . . .                      | 38 |
| MAP_Fit . . . . .                      | 38 |
| math_functions . . . . .               | 43 |
| MCMC_Fit . . . . .                     | 45 |
| parameters_code . . . . .              | 53 |

|                                    |     |
|------------------------------------|-----|
| parameter_types . . . . .          | 54  |
| plot.ce_rtmb . . . . .             | 55  |
| plot.rtmb_lsmeans . . . . .        | 55  |
| plot_acf . . . . .                 | 56  |
| plot_conditional_effects . . . . . | 56  |
| plot_dens . . . . .                | 57  |
| plot_forest . . . . .              | 58  |
| plot_item_curve . . . . .          | 58  |
| plot_item_info . . . . .           | 59  |
| plot_lsmeans . . . . .             | 59  |
| plot_mdu . . . . .                 | 60  |
| plot_pairs . . . . .               | 61  |
| plot_test_info . . . . .           | 62  |
| plot_trace . . . . .               | 62  |
| posterior_predict . . . . .        | 63  |
| pp_check . . . . .                 | 63  |
| print.bayes_factor . . . . .       | 65  |
| print.bayes_factor_rtmb . . . . .  | 65  |
| print.ce_rtmb . . . . .            | 66  |
| print.ce_simple . . . . .          | 66  |
| print.summary_BayesRTMB . . . . .  | 67  |
| prior_flat . . . . .               | 67  |
| prior_jzs . . . . .                | 68  |
| prior_normal . . . . .             | 68  |
| prior_rhs . . . . .                | 69  |
| prior_ssp . . . . .                | 70  |
| prior_uniform . . . . .            | 70  |
| prior_weak . . . . .               | 71  |
| quad_form_chol . . . . .           | 71  |
| quad_form_diag . . . . .           | 72  |
| quantile95 . . . . .               | 72  |
| read_mcmc_csv . . . . .            | 73  |
| rhat_summary . . . . .             | 73  |
| rtmb_array . . . . .               | 74  |
| rtmb_code . . . . .                | 75  |
| rtmb_corr . . . . .                | 77  |
| rtmb_fa . . . . .                  | 78  |
| RTMB_Fit_Base . . . . .            | 79  |
| rtmb_glm . . . . .                 | 84  |
| rtmb_glmer . . . . .               | 87  |
| rtmb_irt . . . . .                 | 90  |
| rtmb_lm . . . . .                  | 92  |
| rtmb_lmer . . . . .                | 94  |
| rtmb_loglinear . . . . .           | 97  |
| rtmb_lrt . . . . .                 | 98  |
| rtmb_mdu . . . . .                 | 100 |
| rtmb_mediation . . . . .           | 102 |
| rtmb_mixture . . . . .             | 104 |

|                                   |     |
|-----------------------------------|-----|
| rtmb_model . . . . .              | 105 |
| RTMB_Model-class . . . . .        | 108 |
| rtmb_syntax . . . . .             | 119 |
| rtmb_table . . . . .              | 120 |
| rtmb_ttest . . . . .              | 121 |
| rtmb_vector . . . . .             | 122 |
| rtmb_wrappers . . . . .           | 123 |
| r_hat . . . . .                   | 124 |
| safe_rtmb_model . . . . .         | 125 |
| simple_effects . . . . .          | 125 |
| simple_effects.mcmc_fit . . . . . | 126 |
| softmax . . . . .                 | 127 |
| sort_loadings . . . . .           | 127 |
| squared_distance . . . . .        | 128 |
| stz_basis . . . . .               | 128 |
| summary.ce_rtmb . . . . .         | 129 |
| summary_mcmc . . . . .            | 129 |
| sum_to_zero . . . . .             | 130 |
| test_info . . . . .               | 130 |
| to_centered_matrix . . . . .      | 131 |
| to_centered_tri . . . . .         | 131 |
| to_long . . . . .                 | 132 |
| to_lower_tri . . . . .            | 133 |
| to_wide . . . . .                 | 133 |
| training . . . . .                | 134 |
| transform_code . . . . .          | 134 |
| upgrade_fit . . . . .             | 135 |
| validate_data . . . . .           | 136 |
| VB_Fit . . . . .                  | 136 |

## Index 143

---

|             |                                                               |
|-------------|---------------------------------------------------------------|
| ADVI_method | <i>Automatic Differentiation Variational Inference (ADVI)</i> |
|-------------|---------------------------------------------------------------|

---

### Description

Automatic Differentiation Variational Inference (ADVI)

### Usage

```
ADVI_method(
  model,
  par_list,
  pl_full,
  iter = 3000,
  tol_rel_obj = 0.001,
  window_size = 100,
```

```

    num_samples = 1000,
    alpha = 0.01,
    laplace = FALSE,
    print_freq = 500,
    method = c("meanfield", "fullrank", "hybrid"),
    update_progress = NULL,
    update_interval = 100
  )

```

### Arguments

|                              |                                                                                                                   |
|------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>model</code>           | An RTMB objective function object ('ad_obj').                                                                     |
| <code>par_list</code>        | A list defining the structure of parameters to be estimated.                                                      |
| <code>pl_full</code>         | A list defining the full structure of parameters including random effects.                                        |
| <code>iter</code>            | Integer; fixed number of iterations for the optimization. Default is 3000.                                        |
| <code>tol_rel_obj</code>     | Numeric; relative tolerance for the ELBO to check convergence. Default is 0.001.                                  |
| <code>window_size</code>     | Integer; size of the moving window to calculate the median ELBO. Default is 100.                                  |
| <code>num_samples</code>     | Integer; number of posterior draws to generate after optimization. Default is 1000.                               |
| <code>alpha</code>           | Numeric; learning rate (step size) for the Adam optimizer. Default is 0.01.                                       |
| <code>laplace</code>         | Logical; whether Laplace approximation is used. Default is FALSE.                                                 |
| <code>print_freq</code>      | Integer; frequency of printing progress to the console. Set to 0 to disable. Default is 500.                      |
| <code>method</code>          | Character; type of variational distribution. One of "meanfield", "fullrank", or "hybrid". Default is "meanfield". |
| <code>update_progress</code> | Optional function for progress reporting.                                                                         |
| <code>update_interval</code> | Integer; interval for progress updates. Default is 100.                                                           |

### Value

A list containing 'fit', 'random\_fit', 'elbo\_history', 'elbo\_final', 'rel\_obj\_final', and 'converged'.

---

|                           |                               |
|---------------------------|-------------------------------|
| <code>bayes_factor</code> | <i>Calculate Bayes Factor</i> |
|---------------------------|-------------------------------|

---

### Description

Compare two models by calculating the Bayes Factor based on their marginal likelihoods.

Usage

```
bayes_factor(logm11, logm12, error_threshold = 0.2)
```

Arguments

- logm11            The first model fit (e.g., ‘mcmc\_fit’) or its log-marginal likelihood.
- logm12            The second model fit (e.g., ‘mcmc\_fit’) or its log-marginal likelihood.
- error\_threshold        Threshold for warning about high bridge-sampling error (default 0.2).

Value

An object of class ‘bayes\_factor’ containing the Bayes factor, log Bayes factor, approximate estimation error, and interpretation.

Examples

```
# Compare two models using Bayes Factor
data(debate, package = "BayesRTMB")
fit1 <- rtmb_lm(sat ~ talk, data = debate)
fit2 <- rtmb_lm(sat ~ talk + perf, data = debate)
map1 <- fit1$optimize()
map2 <- fit2$optimize()
bf <- bayes_factor(map1, map2)
print(bf)
```

---

|          |                                 |
|----------|---------------------------------|
| beverage | <i>Beverage Preference Data</i> |
|----------|---------------------------------|

---

Description

A dataset containing preference ratings for various beverages. This data was used to demonstrate a random effect model in metric multidimensional unfolding.

Usage

```
beverage
```

Format

A data frame (or matrix) with [INSERT NUMBER OF ROWS] rows and [INSERT NUMBER OF COLUMNS] columns.

- AJ** Apple Juice preference rating
- BT** Black Tea preference rating
- CF** Coffee preference rating

- CL** Cola preference rating
- GT** Green Tea preference rating
- OJ** Orange Juice preference rating
- OT** Oolong Tea preference rating

**Source**

Adachi, K. (2000). A Random Effect Model in Metric Multidimensional Unfolding. *The Japanese Journal of Behaviormetrics*, 27(1), 12-23. doi:[10.2333/jbhmk.27.12](https://doi.org/10.2333/jbhmk.27.12)

---

|         |                                         |
|---------|-----------------------------------------|
| BigFive | <i>Big Five Personality Traits Data</i> |
|---------|-----------------------------------------|

---

**Description**

A dataset containing responses to 20 items measuring the Big Five personality traits. This dataset was originally collected by the package author. The items are arranged in a cyclical order representing the five factors: Extraversion, Neuroticism, Openness, Agreeableness, and Conscientiousness.

**Usage**

BigFive

**Format**

A data frame with 10 rows and 20 columns:

- BF1** Extraversion item 1
- BF2** Neuroticism item 1
- BF3** Openness item 1
- BF4** Agreeableness item 1
- BF5** Conscientiousness item 1
- BF6** Extraversion item 2
- BF7** Neuroticism item 2
- BF8** Openness item 2
- BF9** Agreeableness item 2
- BF10** Conscientiousness item 2
- BF11** Extraversion item 3
- BF12** Neuroticism item 3
- BF13** Openness item 3
- BF14** Agreeableness item 3
- BF15** Conscientiousness item 3

**BF16** Extraversion item 4

**BF17** Neuroticism item 4

**BF18** Openness item 4

**BF19** Agreeableness item 4

**BF20** Conscientiousness item 4

## Source

Originally collected by the package author.

---

|                   |                                                        |
|-------------------|--------------------------------------------------------|
| center_grand_mean | <i>Center a Numeric Variable Around Its Grand Mean</i> |
|-------------------|--------------------------------------------------------|

---

## Description

Subtracts the overall mean from a numeric vector. This helper is used by the regression wrappers in their generated ‘setup’ code and can also be used in hand-written ‘rtmb\_code()’ models.

## Usage

```
center_grand_mean(x, na.rm = TRUE)
```

## Arguments

`x` Numeric vector to center.

`na.rm` Logical; whether missing values are removed when calculating the mean.

## Value

A numeric vector with the same length as ‘x’.

## Examples

```
center_grand_mean(c(1, 2, 3))
```

---

center\_within\_cluster *Center a Numeric Variable Within Clusters*


---

**Description**

Subtracts each cluster mean from a numeric vector. Missing cluster values produce missing centered values. This helper is used by the regression wrappers in their generated ‘setup’ code and can also be used in hand-written ‘rtmb\_code()’ models.

**Usage**

```
center_within_cluster(x, cluster, na.rm = TRUE)
```

**Arguments**

|         |                                                                             |
|---------|-----------------------------------------------------------------------------|
| x       | Numeric vector to center.                                                   |
| cluster | Vector identifying the cluster for each element of ‘x’.                     |
| na.rm   | Logical; whether missing values are removed when calculating cluster means. |

**Value**

A numeric vector with the same length as ‘x’.

**Examples**

```
center_within_cluster(c(1, 3, 2, 6), c("a", "a", "b", "b"))
```

---

Classic\_Fit *Classic fit object*


---

**Description**

An R6 class representing the results of a classical (frequentist) estimation.

**Super class**

```
BayesRTMB::RTMB_Fit_Base -> Classic_Fit
```

**Public fields**

**model** The 'RTMB\_Model' object used for estimation.  
**fit** The result of the estimation (dataframe or lm object).  
**par** a named list of parameter estimates.  
**vcov** Variance-covariance matrix of fixed effects.  
**se\_method** Character string specifying the method used for standard errors.  
**cluster** Character string specifying the cluster variable name, if any.  
**robust\_type** Character string specifying the robust standard error correction type, if used.  
**bootstrap\_results** A matrix containing bootstrap samples, if applicable.  
**test\_results** List of additional test results (e.g., `chisq.test`).  
**view** Character vector of parameter names to prioritize in summary.  
**par\_vec** Numeric vector of parameter estimates.  
**objective** Final objective value.  
**log\_lik** Log-likelihood.  
**restricted\_log\_lik** Restricted log-likelihood, if classical estimation used REML-style marginalization.  
**rss** Residual sum of squares for classical linear models, if available.  
**df\_residual** Residual degrees of freedom for classical linear models, if available.  
**convergence** Convergence code.  
**sd\_rep** TMB `sdreport` object.  
**df\_fixed** Dataframe of fixed effects results.  
**random\_effects** Dataframe of random effects results.  
**df\_transform** Dataframe of transformed parameters.  
**df\_generate** Dataframe of generated quantities.  
**opt\_history** Dataframe of optimization history.  
**transform** List of transformed parameters.  
**generate** List of generated quantities.  
**se\_samples** List of simulated samples for SE estimation.  
**par\_unc** Numeric vector of unconstrained parameter estimates.  
**vcov\_unc** Variance-covariance matrix of parameters in unconstrained space.  
**ci\_method** Method used for CI estimation.  
**laplace** Whether Laplace approximation was used.  
**map** Parameter mapping used.  
**df\_method** Character string specifying the degrees of freedom calculation method.  
**idx\_fix\_active** Numeric vector; mapping between active parameters and full unconstrained vector.  
**show\_df** Logical; whether to display degrees of freedom in the summary output.

**Methods****Public methods:**

- `Classic_Fit$get_point_estimate()`
- `Classic_Fit$new()`
- `Classic_Fit$robust_se()`
- `Classic_Fit$compute_robust()`
- `Classic_Fit$bootstrap()`
- `Classic_Fit$compute_bootstrap()`
- `Classic_Fit$AIC()`
- `Classic_Fit$BIC()`
- `Classic_Fit$WAIC()`
- `Classic_Fit$diagnose()`
- `Classic_Fit$print()`
- `Classic_Fit$logLik()`
- `Classic_Fit$EAP()`
- `Classic_Fit$MAP()`
- `Classic_Fit$summary()`
- `Classic_Fit$anova()`
- `Classic_Fit$lsmeans()`
- `Classic_Fit$.calc_contrast()`
- `Classic_Fit$.get_lsmeans_df()`
- `Classic_Fit$.construct_par_list()`
- `Classic_Fit$clone()`

**Method** `get_point_estimate()`: Get point estimate for a target parameter.

*Usage:*

```
Classic_Fit$get_point_estimate(target, ...)
```

*Arguments:*

`target` Target parameter name.

`...` Additional arguments, ignored for classic fits.

*Returns:* Matrix, array, vector, or scalar point estimate.

**Method** `new()`: Create a new 'Classic\_Fit' object.

*Usage:*

```
Classic_Fit$new(  
  model,  
  par_vec = NULL,  
  par = NULL,  
  objective = NULL,  
  log_lik = NULL,  
  restricted_log_lik = NULL,  
  convergence = NULL,  
  sd_rep = NULL,
```

```

df_fixed = NULL,
random_effects = NULL,
df_transform = NULL,
df_generate = NULL,
opt_history = NULL,
transform = NULL,
generate = NULL,
se_samples = NULL,
par_unc = NULL,
vcov_unc = NULL,
ci_method = "wald",
laplace = TRUE,
map = NULL,
test_results = list(),
view = NULL,
fit = NULL,
vcov = NULL,
df_method = "auto",
idx_fix_active = NULL,
show_df = TRUE,
rss = NULL,
df_residual = NULL,
...
)

```

*Arguments:*

`model` The 'RTMB\_Model' object.

`par_vec` Numeric vector of parameter estimates.

`par` List of parameter estimates.

`objective` Final objective value.

`log_lik` Full log-likelihood used for information criteria.

`restricted_log_lik` Restricted log-likelihood, if classical estimation used REML-style marginalization.

`convergence` Convergence code.

`sd_rep` TMB sdreport object.

`df_fixed` Dataframe of fixed effects results.

`random_effects` Dataframe of random effects results.

`df_transform` Dataframe of transformed parameters.

`df_generate` Dataframe of generated quantities.

`opt_history` Dataframe of optimization history.

`transform` List of transformed parameters.

`generate` List of generated quantities.

`se_samples` List of simulated samples for SE estimation.

`par_unc` Parameter vector on unconstrained scale.

`vcov_unc` Covariance matrix on unconstrained scale.

`ci_method` Method used for CI estimation.

laplace Whether Laplace approximation was used.  
 map Parameter mapping used.  
 test\_results List of additional test results (e.g., `chisq.test`).  
 view Character vector of parameter names to prioritize in summary.  
 fit Legacy argument for backward compatibility (maps to `df_fixed`).  
 vcov Variance-covariance matrix of parameters.  
 df\_method Method for degrees of freedom calculation.  
 idx\_fix\_active Numeric vector; mapping between active parameters and full unconstrained vector.  
 show\_df Logical; whether to display degrees of freedom in the summary output.  
 rss Residual sum of squares for classical linear models, if available.  
 df\_residual Residual degrees of freedom for classical linear models, if available.  
 ... Additional arguments passed to the constructor.

**Method** `robust_se()`: Compute robust standard errors (sandwich estimator).

*Usage:*

```
Classic_Fit$robust_se(
  cluster = NULL,
  type = c("HC3", "HC0", "HC1", "CR1", "CR0"),
  inplace = FALSE,
  ...
)
```

*Arguments:*

cluster Character; variable name for clustering.  
 type Character; "HC3" (default), "HC0", or "HC1" for non-clustered robust SE; "CR1" or "CR0" for clustered robust SE.  
 inplace Logical; if FALSE, return a robust-SE copy without modifying the original fit. If TRUE, update the object in place.  
 ... Additional arguments.

*Returns:* Self.

**Method** `compute_robust()`: (Deprecated) Use `robust_se()` instead.

*Usage:*

```
Classic_Fit$compute_robust(...)
```

*Arguments:*

... Arguments passed to `robust_se()`, including `inplace`.

*Returns:* Self.

**Method** `bootstrap()`: Compute nonparametric bootstrap standard errors and confidence intervals. Currently implemented only for mediation models. Bootstrap refits mediation equations with base R `lm.fit()/glm.fit()` when possible, and falls back to RTMB refits for unsupported families.

*Usage:*

```
Classic_Fit$bootstrap(n_boot = 1000, seed = NULL, inplace = FALSE, ...)
```

*Arguments:*

`n_boot` Integer; number of bootstrap samples.

`seed` Optional integer random seed.

`inplace` Logical; if FALSE, return a bootstrap-SE copy without modifying the original fit. If TRUE, update the object in place.

... Additional arguments.

*Returns:* A `Classic_Fit` object.

**Method** `compute_bootstrap()`: Compute nonparametric bootstrap standard errors and confidence intervals. Currently implemented only for mediation models. Bootstrap refits mediation equations with base R `lm.fit()/glm.fit()` when possible, and falls back to RTMB refits for unsupported families.

*Usage:*

```
Classic_Fit$compute_bootstrap(n_boot = 1000, seed = NULL, inplace = FALSE, ...)
```

*Arguments:*

`n_boot` Integer; number of bootstrap samples.

`seed` Optional integer random seed.

`inplace` Logical; if FALSE, return a bootstrap-SE copy without modifying the original fit. If TRUE, update the object in place.

... Additional arguments.

*Returns:* A `Classic_Fit` object.

**Method** `AIC()`: Get the AIC of the fitted model.

*Usage:*

```
Classic_Fit$AIC()
```

**Method** `BIC()`: Get the BIC of the fitted model.

*Usage:*

```
Classic_Fit$BIC()
```

**Method** `WAIC()`: WAIC is not defined for classical fits.

*Usage:*

```
Classic_Fit$WAIC()
```

**Method** `diagnose()`: Run basic diagnostics for the classical fit.

*Usage:*

```
Classic_Fit$diagnose(...)
```

*Arguments:*

... Additional arguments passed to `'diagnose_classic_fit()'`.

*Returns:* A `'diagnose_BayesRTMB'` object.

**Method** `print()`: Print the fit results.

*Usage:*

```
Classic_Fit$print(...)
```

*Arguments:*

... Additional arguments passed to 'summary()'.

**Method logLik():** Get the Log-Likelihood of the fitted model.

*Usage:*

```
Classic_Fit$logLik()
```

**Method EAP():** EAP estimates are not available for Classic\_Fit.

*Usage:*

```
Classic_Fit$EAP(...)
```

*Arguments:*

... Ignored.

**Method MAP():** MAP estimates are not available for Classic\_Fit.

*Usage:*

```
Classic_Fit$MAP(...)
```

*Arguments:*

... Ignored.

**Method summary():** Display a summary of the estimation results.

*Usage:*

```
Classic_Fit$summary(view = NULL, digits = 5, max_rows = 10, ranef = FALSE)
```

*Arguments:*

view Character vector of parameter names to prioritize or filter by.

digits Number of digits to print for estimates.

max\_rows Maximum number of rows to display in the coefficient table.

ranef Logical; whether to show random effects in the summary.

**Method anova():** Perform ANOVA (Wald F-tests / Chisq-tests) on the fitted model.

*Usage:*

```
Classic_Fit$anova(method = c("reml", "ls"), type = 3)
```

*Arguments:*

method Character; "reml" (standard) or "ls" (experimental).

type Integer; Type of Sum of Squares (only Type III supported currently).

*Returns:* A data frame containing the ANOVA table.

**Method lsmeans():** Calculate Least Squares Means (Marginal Means) and contrasts.

*Usage:*

```
Classic_Fit$lsmeans(
  specs = NULL,
  pairwise = FALSE,
  simple = NULL,
  adjust = "holm",
  protect = FALSE
)
```

*Arguments:*

`specs` Character vector of factors to calculate means for.  
`pairwise` Logical; whether to perform pairwise comparisons.  
`simple` Character vector of factors to hold constant for simple main effects.  
`adjust` Character; p-value adjustment method (e.g., "bonferroni", "holm", "none").  
`protect` Logical; whether to use hierarchical (protected) testing.

*Returns:* A data frame containing the marginal means or contrasts.

**Method** `.calc_contrast()`: (Internal) Calculate metrics for a contrast.

*Usage:*

```
Classic_Fit$.calc_contrast(L, specs)
```

*Arguments:*

`L` Contrast matrix.  
`specs` Variable names for lsmeans.

*Returns:* A data frame with estimate, SE, df, t-value, and p-value.

**Method** `.get_lsmeans_df()`: (Internal) Get representative DF for lsmeans.

*Usage:*

```
Classic_Fit$.get_lsmeans_df(specs)
```

*Arguments:*

`specs` Variable names for lsmeans.

*Returns:* Degrees of freedom.

**Method** `.construct_par_list()`: (Internal) Construct a list of parameters from the fit.

*Usage:*

```
Classic_Fit$.construct_par_list(fit)
```

*Arguments:*

`fit` The fit result (dataframe or lm object).

*Returns:* A named list of parameters.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
Classic_Fit$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

---

conditional\_effects     *Calculate Conditional Effects*


---

## Description

Calculate and visualize the predicted values (marginal effects) of a variable, potentially conditional on the levels of another variable (interaction).

## Usage

```
conditional_effects(
  fit,
  effect,
  prob = 0.95,
  sd_multiplier = 1,
  sd_slice = NULL,
  ...
)
```

## Arguments

|                            |                                                                                                                                                                                                                                                        |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fit</code>           | Model fit object (e.g., 'mcmc_fit', 'map_fit').                                                                                                                                                                                                        |
| <code>effect</code>        | Name of the variable to visualize (e.g., "X1" or "X1:X2").                                                                                                                                                                                             |
| <code>prob</code>          | Probability for the credible/confidence interval (default is 0.95).                                                                                                                                                                                    |
| <code>sd_multiplier</code> | Numeric. Multiplier for standard deviation when splitting continuous moderators (default is 1).                                                                                                                                                        |
| <code>sd_slice</code>      | Logical or NULL. If TRUE, continuous moderators are evaluated at mean - SD, mean, and mean + SD. If FALSE, all observed moderator values are used. If NULL (default), sd slicing is used automatically when the moderator has 6 or more unique values. |
| <code>...</code>           | Additional arguments.                                                                                                                                                                                                                                  |

## Value

An object of class 'ce\_rtmb' containing the predicted values and their credible intervals.

## Examples

```
data(debate, package = "BayesRTMB")
fit <- rtmb_lm(sat ~ talk * perf, data = debate)
map_fit <- fit$optimize()
ce <- conditional_effects(map_fit, effect = "talk:perf")
plot(ce)
summary(ce)
```

---

conditional\_effects.mcmc\_fit

*Calculate conditional effects for MCMC fit objects*


---

## Description

Calculate conditional effects for MCMC fit objects

## Usage

```
## S3 method for class 'mcmc_fit'
conditional_effects(
  fit,
  effect,
  prob = 0.95,
  sd_multiplier = 1,
  sd_slice = NULL,
  resolution = 100,
  ...
)
```

## Arguments

|               |                                                                                                                                                                                                                                                        |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fit           | An object of class 'MCMC_Fit'.                                                                                                                                                                                                                         |
| effect        | Name of the explanatory variable to visualize (e.g., "X1" or "X1:X2").                                                                                                                                                                                 |
| prob          | Probability for the credible/confidence interval (default is 0.95).                                                                                                                                                                                    |
| sd_multiplier | Numeric. Multiplier for standard deviation when splitting continuous moderators (default is 1).                                                                                                                                                        |
| sd_slice      | Logical or NULL. If TRUE, continuous moderators are evaluated at mean - SD, mean, and mean + SD. If FALSE, all observed moderator values are used. If NULL (default), sd slicing is used automatically when the moderator has 6 or more unique values. |
| resolution    | Grid resolution to calculate for continuous variables (default is 100).                                                                                                                                                                                |
| ...           | Additional arguments.                                                                                                                                                                                                                                  |

## Value

A 'ce\_rtmb' object.

debate

*Debate Simulation Data***Description**

A simulated dataset containing individual satisfaction scores and related variables from a 3-person group debate experiment. This dummy data was created by the package author to demonstrate hierarchical (multilevel) modeling.

**Usage**

debate

**Format**

A data frame with 300 rows and 6 columns:

**group** Group identifier. Each group consists of 3 members.

**sat** Self-reported satisfaction level of the individual after the debate.

**talk** Amount of talk or utterances by the individual during the debate.

**perf** Evaluated performance of the group debate (group-level variable).

**skill** Evaluated conversation skill of the individual.

**cond** Experimental condition indicator (e.g., 0 = control, 1 = treatment).

**Source**

Simulated dummy data created by the package author.

diffusion\_lpdf

*Diffusion model log-probability density function***Description**

Log-likelihood for a two-boundary drift diffusion model. The first argument is the response time and the second argument is the binary response. Positive response values are treated as upper-boundary responses; zero or negative values are treated as lower-boundary responses.

If  $x$  is a matrix, rows are treated as independent units and columns as repeated trials. In that case, `sum = FALSE` returns one log-likelihood contribution per row. If  $x$  is a vector, `sum = FALSE` returns one contribution per observation.

**Usage**

```
diffusion_lpdf(x, response, alpha, tau, beta, delta, K_diff = 20, sum = TRUE)
```

**Arguments**

|          |                                                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------------------------------|
| x        | Response times. A vector, or a matrix with units in rows and trials in columns.                                                     |
| response | Binary responses with the same shape as x. Positive values indicate the upper boundary.                                             |
| alpha    | Boundary separation. A scalar, row-wise vector, trial-wise vector, or matrix compatible with x.                                     |
| tau      | Non-decision time. A scalar, row-wise vector, trial-wise vector, or matrix compatible with x.                                       |
| beta     | Initial bias as a proportion of the boundary separation. A scalar, row-wise vector, trial-wise vector, or matrix compatible with x. |
| delta    | Drift rate. A scalar, row-wise vector, trial-wise vector, or matrix compatible with x.                                              |
| K_diff   | Number of terms in the truncated infinite-series approximation.                                                                     |
| sum      | Logical; if TRUE (default), returns the summed log-likelihood.                                                                      |

**Value**

The summed log-likelihood, row-wise log-likelihoods for matrix input, or observation-wise log-likelihoods for vector input.

---

|     |                                              |
|-----|----------------------------------------------|
| Dim | <i>Define parameter dimensions and types</i> |
|-----|----------------------------------------------|

---

**Description**

Define parameter dimensions and types

**Usage**

```
Dim(dim = 1, type = NULL, lower = NULL, upper = NULL, random = FALSE)
```

**Arguments**

|        |                                         |
|--------|-----------------------------------------|
| dim    | Dimensions of the parameter.            |
| type   | Type of the parameter.                  |
| lower  | Lower bound.                            |
| upper  | Upper bound.                            |
| random | Logical; whether it is a random effect. |

---

|          |                           |
|----------|---------------------------|
| distance | <i>Euclidean distance</i> |
|----------|---------------------------|

---

**Description**

Euclidean distance

**Usage**

```
distance(x, y, eps = 1e-08)
```

**Arguments**

|     |                                                                |
|-----|----------------------------------------------------------------|
| x   | A numeric vector.                                              |
| y   | A numeric vector.                                              |
| eps | A small value added for numerical stability (default is 1e-8). |

**Value**

The Euclidean distance between x and y.

---

|               |                                                  |
|---------------|--------------------------------------------------|
| distributions | <i>Probability Distributions for RTMB Models</i> |
|---------------|--------------------------------------------------|

---

**Description**

This page summarizes the probability distributions available within the ‘rtmb\_code’ block. These functions are designed to be used with the Stan-like sampling syntax (‘~’), which internally adds the log-density to the model’s total log-posterior.

**Details**

**Syntax Styles:** In ‘rtmb\_code’, you can specify distributions in two ways:

- **Sampling Syntax (Recommended):** `y ~ normal(mu, sigma)`
- **Explicit Function Call:** `lp <- lp + normal_lpdf(y, mu, sigma)`

**Continuous Distributions (LPDF):**

- `normal(mean, sd)`: Normal distribution.
- `lognormal(meanlog, sdlog)`: Lognormal distribution.
- `exp_mod_normal(mu, sigma, lambda)`: Exponentially modified normal distribution.
- `exponential(rate)`: Exponential distribution.
- `cauchy(location, scale)`: Cauchy distribution.

- `student_t(df, mu, sigma)`: Student's t-distribution.
- `gamma(shape, rate)`: Gamma distribution.
- `inverse_gamma(shape, scale)`: Inverse-gamma distribution.
- `beta(a, b)`: Beta distribution.

#### Discrete Distributions (LPMF):

- `bernoulli(prob)` / `bernoulli_logit(eta)`: Binary outcomes.
- `binomial(size, prob)` / `binomial_logit(size, eta)`: Binomial outcomes.
- `poisson(mean)`: Poisson count data.
- `neg_binomial_2(mu, size)`: Negative binomial (mean/dispersion parameterization).
- `ordered_logistic(eta, cutpoints)`: Ordered categorical outcomes.
- `sequential_logistic(eta, cutpoints)`: Sequential ordered categorical outcomes.

#### Multivariate and Matrix Distributions:

- `multi_normal(mean, Sigma)`: Standard multivariate normal distribution.
- `multi_student_t(df, mean, Sigma)`: Multivariate Student's t-distribution.
- `multi_cauchy(mean, Sigma)`: Multivariate Cauchy distribution (Student-t with  $df=1$ ).
- `gaussian_process(x, mean, magnitude, smoothing, noise)`: Gaussian process prior with a squared exponential kernel.
- `lkj_corr(eta)`: LKJ prior for correlation matrices.
- `dirichlet(alpha)`: Dirichlet distribution for simplexes.
- `lower_tri_normal(mean, sd)`: Normal distribution for elements of a lower-triangular matrix.
- `centered_tri_multi_normal(sigma)`: Multivariate normal for centered triangular matrices (used in identification constraints).
- `sufficient_multi_normal_fa(S_mat, N, y_bar, mu, psi, Lambda)`: Factor analysis likelihood using sufficient statistics (highly efficient for large sample sizes).

#### Response-time Distributions:

- `diffusion(alpha, tau, beta, delta)`: Drift diffusion model likelihood for response times and binary choices. Use with `obs(RT, Choice) ~ diffusion(...)`.

**Vectorization:** Most univariate distributions are vectorized. If  $y$  and  $\mu$  are vectors, `y ~ normal(mu, sigma)` will calculate the sum of log-densities for all elements efficiently.

---

`ess_basic`*Basic Effective Sample Size for a single chain or pooled chains*

---

**Description**

Basic Effective Sample Size for a single chain or pooled chains

**Usage**

```
ess_basic(sims)
```

**Arguments**

`sims` A numeric vector or matrix of samples.

**Value**

A numeric value.

---

`ess_bulk`*Calculate Bulk Effective Sample Size*

---

**Description**

Calculate Bulk Effective Sample Size

**Usage**

```
ess_bulk(sims)
```

**Arguments**

`sims` A matrix of samples (iterations x chains).

**Value**

A numeric value.

---

|            |                                                                           |
|------------|---------------------------------------------------------------------------|
| ess_tail95 | <i>Calculate Tail Effective Sample Size (at 2.5% and 97.5% quantiles)</i> |
|------------|---------------------------------------------------------------------------|

---

**Description**

Calculate Tail Effective Sample Size (at 2.5% and 97.5% quantiles)

**Usage**

```
ess_tail95(sims)
```

**Arguments**

|      |                                            |
|------|--------------------------------------------|
| sims | A matrix of samples (iterations x chains). |
|------|--------------------------------------------|

**Value**

A numeric value.

---

|                     |                                                                       |
|---------------------|-----------------------------------------------------------------------|
| exp_mod_normal_lpdf | <i>Exponentially modified normal log-probability density function</i> |
|---------------------|-----------------------------------------------------------------------|

---

**Description**

Log-density for  $X = Z + E$ , where  $Z$  follows a normal distribution with mean  $\mu$  and standard deviation  $\sigma$ , and  $E$  follows an exponential distribution with rate  $\lambda$ .

**Usage**

```
exp_mod_normal_lpdf(x, mu, sigma, lambda, sum = TRUE)
```

**Arguments**

|        |                                                                                                             |
|--------|-------------------------------------------------------------------------------------------------------------|
| x      | Vector of quantiles.                                                                                        |
| mu     | Mean of the normal component.                                                                               |
| sigma  | Standard deviation of the normal component.                                                                 |
| lambda | Rate of the exponential component.                                                                          |
| sum    | Logical; if TRUE (default), returns the sum of log-densities. If FALSE, returns element-wise log-densities. |

**Value**

The sum of the log-density or a vector of log-densities.

---

|      |                                       |
|------|---------------------------------------|
| fabs | <i>Smooth absolute value function</i> |
|------|---------------------------------------|

---

**Description**

Smooth absolute value function

**Usage**

```
fabs(x, eps = 1e-08)
```

**Arguments**

|     |                                                                |
|-----|----------------------------------------------------------------|
| x   | A numeric vector.                                              |
| eps | A small value added for numerical stability (default is 1e-8). |

**Value**

The smooth absolute value of x.

---

|                       |                                                                  |
|-----------------------|------------------------------------------------------------------|
| gaussian_process_lpdf | <i>Gaussian Process Log-Density (Squared Exponential Kernel)</i> |
|-----------------------|------------------------------------------------------------------|

---

**Description**

Calculates the log-density of a Gaussian Process with a Squared Exponential (RBF) kernel.

**Usage**

```
gaussian_process_lpdf(  
  y,  
  x,  
  mean = 0,  
  magnitude = 1,  
  smoothing = 1,  
  noise = 0.01,  
  sum = TRUE  
)
```

**Arguments**

|           |                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------|
| y         | Observation vector (N), or matrix (M x N) whose rows are independent GP realizations on the same coordinates.     |
| x         | Coordinate vector or matrix (N x D), where N matches length(y) for vector y or ncol(y) for matrix y.              |
| mean      | Mean vector (scalar, length N, or M x N matrix for matrix y).                                                     |
| magnitude | Signal standard deviation (alpha).                                                                                |
| smoothing | Length-scale (rho).                                                                                               |
| noise     | Measurement noise standard deviation (sigma).                                                                     |
| sum       | Logical; whether to return the sum of log-densities. If FALSE and y is a matrix, returns one log-density per row. |

**Value**

Log-density value.

---

generate\_random\_init    *Generate Random Initial Values*

---

**Description**

Generates random initial values for model parameters by drawing from a uniform distribution on the unconstrained scale and then transforming them to the constrained scale.

**Usage**

```
generate_random_init(pl_full, par_list, range = 2)
```

**Arguments**

|          |                                                                                                                           |
|----------|---------------------------------------------------------------------------------------------------------------------------|
| pl_full  | A list containing the full parameter structure, including the 'init' field which serves as a template.                    |
| par_list | A list defining the structure and constraints of the parameters, including 'unc_length'.                                  |
| range    | Numeric; the range for the uniform distribution '[-range, range]' used for generating unconstrained values. Default is 2. |

**Value**

A numeric vector of initial values where 'NA' elements are replaced with randomly generated constrained values.

---

|           |                               |
|-----------|-------------------------------|
| inv_logit | <i>Inverse logit function</i> |
|-----------|-------------------------------|

---

**Description**

Inverse logit function

**Usage**

```
inv_logit(x)
```

**Arguments**

|   |                   |
|---|-------------------|
| x | A numeric vector. |
|---|-------------------|

**Value**

The inverse logit of x.

---

|            |                                                                |
|------------|----------------------------------------------------------------|
| item_curve | <i>Calculate Item Response Curve / Category Response Curve</i> |
|------------|----------------------------------------------------------------|

---

**Description**

Calculate Item Response Curve / Category Response Curve

**Usage**

```
item_curve(x, ...)
```

**Arguments**

|     |                       |
|-----|-----------------------|
| x   | An object of class    |
| ... | Additional arguments. |

---

```
item_curve.RTMB_Fit_Base
```

*Item Response Curve for RTMB\_Fit\_Base*

---

### Description

Item Response Curve for RTMB\_Fit\_Base

### Usage

```
## S3 method for class 'RTMB_Fit_Base'
item_curve(x, theta_seq = seq(-4, 4, length.out = 100), items = NULL, ...)
```

### Arguments

|           |                                                                               |
|-----------|-------------------------------------------------------------------------------|
| x         | An object of class RTMB_Fit_Base.                                             |
| theta_seq | Sequence of trait values (ability) to evaluate.                               |
| items     | Index or item names to restrict the calculation to specific items (optional). |
| ...       | Additional arguments.                                                         |

### Examples

```
fit <- rtmb_irt(data = BigFive[, 1:5], model = "2PL", type = "ordered")
map_fit <- fit$optimize()
ic <- item_curve(map_fit)
plot(ic)
```

---

```
item_info
```

*Calculate Item Information Function*

---

### Description

Calculate Item Information Function

### Usage

```
item_info(x, ...)
```

### Arguments

|     |                                  |
|-----|----------------------------------|
| x   | An object of class RTMB_Fit_Base |
| ... | Additional arguments.            |

**Examples**

```
fit <- rtmb_irt(data = BigFive[, 1:5], model = "2PL", type = "ordered")
map_fit <- fit$optimize()
ii <- item_info(map_fit)
plot(ii)
```

---

|                                                    |
|----------------------------------------------------|
| item_info.RTMB_Fit_Base                            |
| <i>Item Information Function for RTMB_Fit_Base</i> |

---

**Description**

Item Information Function for RTMB\_Fit\_Base

**Usage**

```
## S3 method for class 'RTMB_Fit_Base'
item_info(x, theta_seq = seq(-4, 4, length.out = 100), items = NULL, ...)
```

**Arguments**

- |           |                                                                               |
|-----------|-------------------------------------------------------------------------------|
| x         | An object of class RTMB_Fit_Base.                                             |
| theta_seq | Sequence of trait values (ability) to evaluate.                               |
| items     | Index or item names to restrict the calculation to specific items (optional). |
| ...       | Additional arguments.                                                         |

---

|       |                           |
|-------|---------------------------|
| log1m | <i>Log of one minus x</i> |
|-------|---------------------------|

---

**Description**

Log of one minus x

**Usage**

```
log1m(x)
```

**Arguments**

- |   |                   |
|---|-------------------|
| x | A numeric vector. |
|---|-------------------|

**Value**

The natural logarithm of 1 - x.

---

|           |                                          |
|-----------|------------------------------------------|
| log1m_exp | <i>Log of one minus exponential of x</i> |
|-----------|------------------------------------------|

---

**Description**

Log of one minus exponential of x

**Usage**

log1m\_exp(x)

**Arguments**

x                      A numeric vector.

**Value**

The natural logarithm of  $1 - \exp(x)$ .

---

|           |                                         |
|-----------|-----------------------------------------|
| log1p_exp | <i>Log of one plus exponential of x</i> |
|-----------|-----------------------------------------|

---

**Description**

Log of one plus exponential of x

**Usage**

log1p\_exp(x)

**Arguments**

x                      A numeric vector.

**Value**

The natural logarithm of  $1 + \exp(x)$ .

---

`logit`*Logit function*

---

**Description**

Logit function

**Usage**

```
logit(x)
```

**Arguments**

`x` A numeric vector of probabilities.

**Value**

The logit of `x`.

---

`log_det_chol`*Log determinant of a Cholesky factor*

---

**Description**

Log determinant of a Cholesky factor

**Usage**

```
log_det_chol(L)
```

**Arguments**

`L` A lower triangular Cholesky factor matrix.

**Value**

The log determinant of the corresponding covariance matrix.

---

|         |                                         |
|---------|-----------------------------------------|
| log_mix | <i>Log mixture of two probabilities</i> |
|---------|-----------------------------------------|

---

**Description**

Log mixture of two probabilities

**Usage**

```
log_mix(theta, lp1, lp2)
```

**Arguments**

|       |                                          |
|-------|------------------------------------------|
| theta | Mixing proportion (between 0 and 1).     |
| lp1   | Log-probability of the first component.  |
| lp2   | Log-probability of the second component. |

**Value**

The log-probability of the mixture.

---

|             |                             |
|-------------|-----------------------------|
| log_softmax | <i>Log-softmax function</i> |
|-------------|-----------------------------|

---

**Description**

Log-softmax function

**Usage**

```
log_softmax(x)
```

**Arguments**

|   |                   |
|---|-------------------|
| x | A numeric vector. |
|---|-------------------|

**Value**

A numeric vector of log-softmax probabilities.

---

|             |                             |
|-------------|-----------------------------|
| log_sum_exp | <i>Log-sum-exp function</i> |
|-------------|-----------------------------|

---

**Description**

Log-sum-exp function

**Usage**

```
log_sum_exp(x, y = NULL)
```

**Arguments**

|   |                             |
|---|-----------------------------|
| x | A numeric vector or scalar. |
| y | An optional numeric scalar. |

**Value**

The log of the sum of the exponentials.

---

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| log_sum_exp_matrix | <i>Log-sum-exp function for matrices (row-wise)</i> |
|--------------------|-----------------------------------------------------|

---

**Description**

Log-sum-exp function for matrices (row-wise)

**Usage**

```
log_sum_exp_matrix(M)
```

**Arguments**

|   |                                      |
|---|--------------------------------------|
| M | A numeric matrix or advector matrix. |
|---|--------------------------------------|

**Value**

A numeric vector of row-wise log-sum-exp.

---

|         |                                             |
|---------|---------------------------------------------|
| lsmeans | <i>Least Squares Means (Marginal Means)</i> |
|---------|---------------------------------------------|

---

**Description**

Calculate least squares means (also known as marginal means or predicted means) for categorical factors in a fitted model. It also supports pairwise comparisons and simple main effects.

**Usage**

```
lsmeans(object, specs, ...)
```

**Arguments**

- object            A fitted model object (e.g., 'Classic\_Fit').
- specs            A character vector of factor names to calculate means for.
- ...               Additional arguments passed to the method.

---

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| make_bw_from_ydif | <i>Make Best and Worst Responses from Best-Worst Pair Indices</i> |
|-------------------|-------------------------------------------------------------------|

---

**Description**

Converts a matrix of Best-Worst pair indices ('Y\_dif') into separate 'Best' and 'Worst' data frames. The returned values are positions within each row of 'sets', not the item labels stored in 'sets'.

**Usage**

```
make_bw_from_ydif(Y_dif, sets)

restore_bw_from_ydif(Y_dif, sets)
```

**Arguments**

- Y\_dif            Matrix or data frame of pair indices (N persons x P tasks). Each value must be an integer from 1 to 'C \* (C - 1)', where 'C' is the number of items per task.
- sets             Matrix or data frame of presented item sets (P tasks x C items).

**Value**

A list with two data frames: 'Best' and 'Worst'.

---

|                     |                                        |
|---------------------|----------------------------------------|
| make_glmer_re_terms | <i>Prepare GLMM Formula Components</i> |
|---------------------|----------------------------------------|

---

## Description

Builds the response, fixed-effect model matrix, and lme4-style random-effect design components used by `'rtmb_glmer()'`. This is the user-facing helper that reproduces what the wrapper places in the generated `'setup'` block.

## Usage

```
make_glmer_re_terms(  
  formula,  
  data = NULL,  
  family = "gaussian",  
  resid_group = NULL,  
  resid_time = NULL,  
  within = NULL,  
  factors = NULL,  
  missing = "listwise",  
  contrasts = NULL  
)
```

## Arguments

|                          |                                                                                                                                                                                                                                       |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>formula</code>     | lme4-style formula.                                                                                                                                                                                                                   |
| <code>data</code>        | Optional data frame. If omitted, variables are resolved from the formula environment. This is useful inside an <code>'rtmb_code()'</code> <code>'setup'</code> block, where data-frame columns are available as individual variables. |
| <code>family</code>      | Character string of the distribution family.                                                                                                                                                                                          |
| <code>resid_group</code> | Optional residual-correlation grouping variable.                                                                                                                                                                                      |
| <code>resid_time</code>  | Optional residual-correlation time variable.                                                                                                                                                                                          |
| <code>within</code>      | Optional list for wide-to-long conversion when the response is written as <code>'cbind(...)'</code> .                                                                                                                                 |
| <code>factors</code>     | Optional character vector of variables to treat as factors.                                                                                                                                                                           |
| <code>missing</code>     | Missing value handling strategy: "listwise".                                                                                                                                                                                          |
| <code>contrasts</code>   | Optional contrast type, either <code>"treatment"</code> or <code>"sum"</code> .                                                                                                                                                       |

## Value

A list containing `'Y'`, `'X'`, `'trials'`, `'offset'`, `'N'`, fixed-effect metadata, and random-effect terms.

---

|                     |                                                                     |
|---------------------|---------------------------------------------------------------------|
| make_glmer_Z_matrix | <i>Reconstruct an Observation-Level Random-Effect Design Matrix</i> |
|---------------------|---------------------------------------------------------------------|

---

### Description

Converts the transposed block random-effect design matrix used internally by ‘rtmb\_glmer()’ into an observation-level matrix. This is kept as a small utility for users who want to work from the block matrix returned by ‘make\_glmer\_re\_terms()’.

### Usage

```
make_glmer_Z_matrix(Zt, group_idx, N = length(group_idx))
```

### Arguments

|           |                                                          |
|-----------|----------------------------------------------------------|
| Zt        | Transposed block random-effect design matrix.            |
| group_idx | Integer group index for each observation.                |
| N         | Number of observations. Defaults to ‘length(group_idx)’. |

### Value

A matrix with ‘N’ rows and one column per random-effect coefficient.

---

|               |                                                             |
|---------------|-------------------------------------------------------------|
| make_init_mdu | <i>Create Initial Values for Multidimensional Unfolding</i> |
|---------------|-------------------------------------------------------------|

---

### Description

Create Initial Values for Multidimensional Unfolding

### Usage

```
make_init_mdu(
  Y,
  D,
  distance = c("squared", "euclidean"),
  alpha = c("random", "fix"),
  distance_eps = 1e-04,
  min_sigma = 0.1
)
```

**Arguments**

|              |                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------|
| Y            | Numeric matrix or data frame (N rows x M items).                                                      |
| D            | Number of unfolding dimensions.                                                                       |
| distance     | Character; "squared" or "euclidean".                                                                  |
| alpha        | Character; "random" for item-specific alpha initial values or "fix" for a common alpha initial value. |
| distance_eps | Small positive constant added to the distance.                                                        |
| min_sigma    | Minimum initial residual standard deviation.                                                          |

**Value**

A named list of initial values.

---

|                   |                                                                   |
|-------------------|-------------------------------------------------------------------|
| make_ydif_from_bw | <i>Make Best-Worst Pair Indices from Best and Worst Responses</i> |
|-------------------|-------------------------------------------------------------------|

---

**Description**

Converts separate 'Best' and 'Worst' response matrices into 'Y\_dif' pair indices for Best-Worst MDU models. The returned index is the position of the ordered '(best, worst)' pair among all 'C \* (C - 1)' position pairs generated from each row of 'sets'.

**Usage**

```
make_ydif_from_bw(Best, Worst, sets)
```

**Arguments**

|       |                                                                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Best  | Matrix or data frame of best responses (N persons x P tasks). Values must be positions within the corresponding row of 'sets', from '1' to 'ncol(sets)'.  |
| Worst | Matrix or data frame of worst responses (N persons x P tasks). Values must be positions within the corresponding row of 'sets', from '1' to 'ncol(sets)'. |
| sets  | Matrix or data frame of presented item sets (P tasks x C items).                                                                                          |

**Value**

An integer matrix of pair indices ('Y\_dif').

|                                     |                                     |
|-------------------------------------|-------------------------------------|
| map_est                             | Maximum A Posteriori (MAP) Estimate |
| <b>Description</b>                  |                                     |
| Maximum A Posteriori (MAP) Estimate |                                     |
| <b>Usage</b>                        |                                     |
| map_est(z)                          |                                     |
| <b>Arguments</b>                    |                                     |
| z                                   | A numeric vector of samples.        |
| <b>Value</b>                        |                                     |
| A numeric value.                    |                                     |
| MAP_Fit                             | MAP fit object                      |

**Description**

MAP fit object  
MAP fit object

**Details**

An R6 class storing optimization results from maximum a posteriori (MAP) estimation.

**Super class**

`BayesRTMB::RTMB_Fit_Base` -> `map_fit`

**Public fields**

- `model` The 'RTMB\_Model' object used for estimation.
- `par_vec` Parameter vector on the unconstrained scale (constrained values unlisted).
- `par` Parameter list on the constrained scale.
- `par_unc` Parameter vector on the unconstrained scale (raw unconstrained values).
- `ci_method` Method used for CI estimation ("wald", "sampling", or "none").
- `objective` RTMB objective function object.
- `log_ml` Log marginal likelihood or related model criterion.

convergence Optimizer convergence code.  
 sd\_rep Standard deviation report object.  
 df\_fixed Summary table for fixed-effect parameters.  
 random\_effects Random effect estimates.  
 df\_transform Summary table for transformed parameter estimates.  
 df\_generate Summary table for generated quantity estimates.  
 opt\_history A vector of optimize objective history.  
 transform List of transformed parameters maintaining their original dimensions.  
 generate List of generated quantities maintaining their original dimensions.  
 se\_samples List of simulated samples for standard error estimation.  
 laplace Logical; whether Laplace approximation was used.  
 vcov\_unc Variance-covariance matrix of parameters in unconstrained space.  
 map List; the parameter mapping used.  
 marginal\_vars Character vector of parameter names requested through 'optimize(marginal = ...)'.  
 laplace\_random\_vars Character vector of all parameter names passed to 'MakeADFun(random = ...)' during Laplace approximation.  
 idx\_fix\_active Numeric vector; mapping between active parameters and full unconstrained vector.  
 show\_df Logical; whether to display degrees of freedom in the summary output.  
 view Character vector of parameter names to prioritize in summary.  
 fallback\_needed Logical; whether Hessian/SE fallback was used during optimization.

## Methods

### Public methods:

- `MAP_Fit$get_point_estimate()`
- `MAP_Fit$new()`
- `MAP_Fit$ranef()`
- `MAP_Fit$draws()`
- `MAP_Fit$summary()`
- `MAP_Fit$print()`
- `MAP_Fit$generated_quantities()`
- `MAP_Fit$WAIC()`
- `MAP_Fit$diagnose()`
- `MAP_Fit$profile()`
- `MAP_Fit$clone()`

**Method** `get_point_estimate()`: Get point estimate for a target parameter.

*Usage:*

```
MAP_Fit$get_point_estimate(target, ...)
```

*Arguments:*

`target` Target parameter name.  
 ... Additional arguments, ignored for MAP fits.  
*Returns:* Matrix, array, vector, or scalar point estimate.

**Method** `new()`: Create a new 'MAP\_Fit' object.

*Usage:*

```
MAP_Fit$new(
  model,
  par_vec = NULL,
  par = NULL,
  objective = NULL,
  log_ml = NULL,
  convergence = NULL,
  sd_rep = NULL,
  df_fixed = NULL,
  random_effects = NULL,
  df_transform = NULL,
  df_generate = NULL,
  opt_history = NULL,
  transform = NULL,
  generate = NULL,
  se_samples = NULL,
  par_unc = NULL,
  ci_method = "wald",
  laplace = TRUE,
  map = NULL,
  vcov_unc = NULL,
  marginal_vars = NULL,
  laplace_random_vars = NULL,
  idx_fix_active = NULL,
  show_df = TRUE,
  view = NULL,
  fallback_needed = NULL
)
```

*Arguments:*

`model` The 'RTMB\_Model' object used for estimation.  
`par_vec` Parameter vector on the unconstrained scale (constrained values unlisted).  
`par` Parameter list on the constrained scale.  
`objective` The objective function value at the optimum.  
`log_ml` Log marginal likelihood.  
`convergence` Optimizer convergence code.  
`sd_rep` The 'sdreport' object from TMB.  
`df_fixed` Data frame of fixed effects estimates and CIs.  
`random_effects` Data frame of random effects estimates and CIs.  
`df_transform` Data frame of transformed parameters.  
`df_generate` Data frame of generated quantities.

opt\_history Data frame of optimization history.  
 transform List of transformed parameters maintaining their original dimensions.  
 generate List of generated quantities maintaining their original dimensions.  
 se\_samples List of simulated samples for standard error estimation.  
 par\_unc Parameter vector on the unconstrained scale (raw values).  
 ci\_method Method used for CI estimation ("wald", "sampling", or "none").  
 laplace Logical; whether Laplace approximation was used.  
 map List; the parameter mapping used.  
 vcov\_unc Variance-covariance matrix of parameters in unconstrained space.  
 marginal\_vars Character vector of parameter names requested through 'optimize(marginal = ...)'.  
 laplace\_random\_vars Character vector of all parameter names passed to 'MakeADFun(random = ...)' during Laplace approximation.  
 idx\_fix\_active Numeric vector; mapping between active parameters and full unconstrained vector.  
 show\_df Logical; whether to display degrees of freedom in the summary output. Default is TRUE.  
 view Character vector of parameter names to prioritize in summary.  
 fallback\_needed Logical; whether Hessian/SE fallback was used during optimization.

**Method** ranef(): Return random effect estimates as a named list.

*Usage:*

```
MAP_Fit$ranef()
```

*Returns:* A named list of random effect estimates.

**Method** draws(): Extract samples from the asymptotic posterior distribution.

*Usage:*

```
MAP_Fit$draws(
  pars = NULL,
  inc_random = FALSE,
  inc_transform = TRUE,
  inc_generate = TRUE,
  ...
)
```

*Arguments:*

pars Character or numeric vector specifying the names or indices of parameters to extract.  
 inc\_random Logical; whether to include random effects.  
 inc\_transform Logical; whether to include transformed parameters.  
 inc\_generate Logical; whether to include generated quantities.  
 ... Ignored.

*Returns:* An array of samples [iterations, 1, parameters].

**Method** summary(): Summarize MAP estimates.

*Usage:*

```
MAP_Fit$summary(
  pars = NULL,
  max_rows = 10,
  digits = 5,
  ranef = FALSE,
  view = NULL
)
```

*Arguments:*

**pars** Character vector specifying the names of parameters to summarize. If NULL, all available parameters are summarized.

**max\_rows** Maximum number of rows to print in summaries. Default is 10.

**digits** Number of digits to print.

**ranef** Logical; whether to also display random effect estimates. Default is FALSE.

**view** Character vector of parameter names to prioritize or filter by.

*Returns:* A summary object, typically a data frame.

**Method print():** Print a brief summary of the fitted object.

*Usage:*

```
MAP_Fit$print(pars = NULL, max_rows = 10, digits = 5, ...)
```

*Arguments:*

**pars** Character vector specifying the names of parameters to summarize.

**max\_rows** Maximum number of rows to print in summaries.

**digits** Number of digits to print.

**...** Additional arguments passed to the ‘summary’ method.

*Returns:* The object itself, invisibly.

**Method generated\_quantities():** Compute generated quantities from the MAP estimate.

*Usage:*

```
MAP_Fit$generated_quantities(code)
```

*Arguments:*

**code** An ‘rtmb\_code({ ... })’ or ‘{ ... }’ block containing the logic to be calculated using the MAP estimate.

*Returns:* The ‘MAP\_Fit’ object itself (invisibly). Results are added or updated in the ‘generate’ list and ‘df\_generate’.

**Method WAIC():** Compute an approximate WAIC from sampling-based uncertainty propagation.

*Usage:*

```
MAP_Fit$WAIC()
```

*Returns:* A ‘waic\_BayesRTMB’ object.

**Method diagnose():** Run basic diagnostics for the MAP fit.

*Usage:*

```
MAP_Fit$diagnose(...)
```

*Arguments:*

... Additional arguments passed to 'diagnose\_map\_fit()'.

*Returns:* A 'diagnose\_BayesRTMB' object.

**Method** `profile()`: Calculate Profile Likelihood confidence intervals for specific parameters.

*Usage:*

```
MAP_Fit$profile(
  pars = NULL,
  level = 0.95,
  trace = FALSE,
  digits = 5,
  show_plot = FALSE,
  quiet = FALSE,
  jacobian = "none",
  ...
)
```

*Arguments:*

`pars` Character vector of parameter names to profile. If NULL, all fixed parameters are profiled.

`level` Confidence level (default is 0.95).

`trace` Logical; whether to print profiling progress. Default is FALSE.

`digits` Integer; number of decimal places to print. Default is 5.

`show_plot` Logical; whether to plot the profile likelihood curves. Default is FALSE.

`quiet` Logical; whether to suppress text output. Default is FALSE.

`jacobian` Character; "none" (default), "random", or "all". Whether to include Jacobian adjustments for transformations.

... Additional arguments passed to `TMB::tmbprofile` (e.g., `ytol`).

*Returns:* A data frame containing the profile-based confidence intervals, with the raw profile objects stored in the "profiles" attribute.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
MAP_Fit$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

**Description**

This page summarizes the utility functions provided to assist in writing efficient and numerically stable model code within 'rtmb\_code'. These functions are specifically designed to be compatible with RTMB's Automatic Differentiation (AD).

## Details

### 1. Link and Inverse Functions:

- `logit(x)`: Computes the logit transformation  $\log(x/(1-x))$ .
- `inv_logit(x)`: Computes the inverse logit (logistic) transformation  $1/(1+\exp(-x))$ .

### 2. Numerical Stability Utilities:

These functions use specialized algorithms to prevent overflow/underflow in AD calculations.

- `log_sum_exp(x)`: Safely computes  $\log(\sum(\exp(x)))$  using the log-sum-exp trick.
- `log1p_exp(x)`: Computes  $\log(1 + \exp(x))$  stably.
- `log1m_exp(x)`: Computes  $\log(1 - \exp(x))$  stably for  $x < 0$ .
- `log_softmax(x)`: Computes the log of the softmax function.
- `fabs(x)`: A smooth version of the absolute value function  $\text{abs}(x)$  to ensure differentiability at zero.

### 3. Matrix and Vector Transformations:

Used to convert unconstrained vectors into structured matrices (e.g., for factor analysis or identification constraints).

- `sum_to_zero(x)`: Transforms a vector of length  $K-1$  into a vector of length  $K$  that sums to zero.
- `to_lower_tri(x, M, D)`: Fills a matrix of size  $M \times D$  with elements from vector  $x$  in a lower-triangular fashion.
- `to_centered_matrix(x, R, C)`: Creates an  $R \times C$  matrix where each column sums to zero.
- `to_centered_tri(x, R, C)`: Creates an  $R \times C$  matrix with column-wise sum-to-zero constraints on the lower elements (useful for identification in factor analysis).
- `rtmb_vector(value, length)`: Creates an AD-compatible vector initialized with `value`.
- `rtmb_array(value, dim)`: Creates an AD-compatible array initialized with `value`.

### 4. Linear Algebra for AD:

- `log_det_chol(L)`: Calculates the log-determinant of a covariance matrix from its Cholesky factor  $L$ .
- `quad_form_chol(x, L)`: Computes the quadratic form  $x^T \Sigma^{-1} x$  using the Cholesky factor  $L$ .
- `distance(x, y)`: Computes the Euclidean distance between two vectors with a small epsilon for stability.

MCMC\_Fit

*MCMC fit object***Description**

MCMC fit object

MCMC fit object

**Details**

An R6 class storing posterior samples and related information from MCMC estimation.

Bayes factors are computed as ratios of marginal likelihoods estimated by bridge sampling. The comparison model can be constructed by fixing parameters with ‘fixed = list(...)’, or supplied as an already fitted ‘MCMC\_Fit’ object via ‘comparison\_fit’. Bayes factors are computed only by marginal-likelihood model comparison.

**Super class**

`BayesRTMB::RTMB_Fit_Base -> mcmc_fit`

**Public fields**

`model` An ‘RTMB\_Model’ object used for estimation.

`fit` Posterior draws for model parameters.

`random_fit` Posterior draws for random effects.

`transform_fit` Posterior draws for transformed parameters.

`transform_dims` Dimension information for transformed parameters.

`generate_fit` Posterior draws for generated quantities.

`generate_dims` Dimension information for generated quantities.

`eps` Step size used by the sampler.

`accept` Acceptance statistics from sampling.

`treedepth` Tree depth used in HMC/NUTS sampling.

`n_leapfrog` Number of leapfrog steps for retained draws.

`divergent` Logical matrix indicating divergent transitions.

`energy` Hamiltonian energy for retained draws.

`metric` Per-chain inverse mass matrix used by NUTS.

`metric_type` Effective mass matrix adaptation type used by NUTS.

`metric_requested` Mass matrix adaptation type requested by the user.

`metric_effective` Effective mass matrix adaptation type by chain.

`metric_auto` Automatic metric selection details by chain, if used.

`metric_init` Initial metric source used by NUTS.

metric\_adaptation Metric learning mode used during warmup.  
nuts\_variant NUTS proposal selection variant used for sampling.  
warmup\_diagnostics Per-chain warmup diagnostics.  
chain\_state Final unconstrained sampler position for each chain.  
sampler\_config Sampling settings required to continue the chains.  
laplace Logical; whether Laplace approximation was used.  
posterior\_mean Posterior mean estimates.  
log\_ml Numeric value storing the calculated log marginal likelihood from bridge sampling.  
comparison\_fit An MCMC\_Fit object containing the fitted comparison model.  
max\_treedepth Maximum tree depth requested for NUTS/HMC.  
pd\_error\_count Positive-definite/singularity errors treated as  $lp = -\text{Inf}$  by chain.

## Methods

### Public methods:

- `MCMC_Fit$get_point_estimate()`
- `MCMC_Fit$new()`
- `MCMC_Fit$print()`
- `MCMC_Fit$continue_sampling()`
- `MCMC_Fit$draws()`
- `MCMC_Fit$summary()`
- `MCMC_Fit$rhat_summary()`
- `MCMC_Fit$unconstrain_draws()`
- `MCMC_Fit$log_prob()`
- `MCMC_Fit$bridgesampling()`
- `MCMC_Fit$WAIC()`
- `MCMC_Fit$diagnose()`
- `MCMC_Fit$bayes_factor()`
- `MCMC_Fit$transformed_draws()`
- `MCMC_Fit$generated_quantities()`
- `MCMC_Fit$resolve_switching()`
- `MCMC_Fit$clone()`

**Method** `get_point_estimate()`: Get point estimate for a target parameter.

*Usage:*

```
MCMC_Fit$get_point_estimate(target, chains = NULL, best_chains = NULL)
```

*Arguments:*

`target` Target parameter name.

`chains` Numeric vector of chains to include. If `NULL`, all chains are used.

`best_chains` Integer; number of best chains to retain based on mean log-posterior.

*Returns:* Matrix, array, vector, or scalar point estimate.

**Method** `new()`: Create a new ‘MCMC\_Fit’ object.

*Usage:*

```
MCMC_Fit$new(
  model,
  fit,
  random_fit,
  eps,
  accept,
  treedepth,
  laplace,
  posterior_mean,
  max_treedepth = NULL,
  pd_error_count = NULL,
  n_leapfrog = NULL,
  divergent = NULL,
  energy = NULL,
  metric = NULL,
  metric_type = NULL,
  metric_init = NULL,
  metric_requested = NULL,
  metric_effective = NULL,
  metric_auto = NULL,
  metric_adaptation = NULL,
  nuts_variant = NULL,
  warmup_diagnostics = NULL,
  chain_state = NULL,
  sampler_config = NULL
)
```

*Arguments:*

`model` An ‘RTMB\_Model’ object used for estimation.  
`fit` Posterior draws for model parameters.  
`random_fit` Posterior draws for random effects, if available.  
`eps` Step size used by the sampler.  
`accept` Acceptance statistics from sampling.  
`treedepth` Tree depth used in HMC/NUTS sampling.  
`laplace` Logical; whether Laplace approximation was used.  
`posterior_mean` Posterior mean estimates.  
`max_treedepth` Maximum tree depth requested for NUTS/HMC.  
`pd_error_count` Positive-definite/singularity errors treated as ‘lp = -Inf’ by chain.  
`n_leapfrog` Number of leapfrog steps for retained draws.  
`divergent` Logical matrix indicating divergent transitions.  
`energy` Hamiltonian energy for retained draws.  
`metric` Per-chain inverse mass matrix used by NUTS.  
`metric_type` Effective mass matrix adaptation type used by NUTS.  
`metric_init` Initial metric source used by NUTS.

metric\_requested Mass matrix adaptation type requested by the user.  
 metric\_effective Effective mass matrix adaptation type by chain.  
 metric\_auto Automatic metric selection details by chain, if used.  
 metric\_adaptation Metric learning mode used during warmup.  
 nuts\_variant NUTS proposal selection variant used for sampling.  
 warmup\_diagnostics Per-chain warmup diagnostics.  
 chain\_state Final unconstrained sampler position for each chain.  
 sampler\_config Sampling settings required to continue the chains.

**Method** print(): Print a brief summary of the fitted object.

*Usage:*

```
MCMC_Fit$print(...)
```

*Arguments:*

... Additional arguments.

*Returns:* The object itself, invisibly.

**Method** continue\_sampling(): Continue all NUTS chains from their saved final states.

The adapted mass matrix and step size from each chain are reused, so no additional warmup is performed. New posterior draws and diagnostics are appended along the iteration dimension.

*Usage:*

```
MCMC_Fit$continue_sampling(
  sampling = 1000,
  thin = NULL,
  seed = sample.int(1e+06, 1),
  max_treedepth = NULL,
  parallel = FALSE,
  inplace = TRUE,
  globals = FALSE,
  progress = c("auto", "none", "bar", "message")
)
```

*Arguments:*

sampling Positive integer; number of additional post-warmup NUTS transitions per chain before thinning. Default is 1000.

thin Optional positive integer thinning interval. 'NULL' reuses the interval from the original sampling run.

seed Integer random seed.

max\_treedepth Optional positive integer maximum tree depth. 'NULL' reuses the value from the original sampling run.

parallel Logical; whether to continue chains in parallel.

inplace Logical; if 'TRUE', append draws to this object. If 'FALSE', return an independently cloned and extended object.

globals Logical; if 'TRUE', let 'future' discover globals when 'parallel = TRUE'.

progress Progress reporting style: "auto", "none", "bar", or "message".

*Returns:* The extended 'MCMC\_Fit' object, invisibly when 'inplace = TRUE'.

**Method** `draws()`: Extract posterior draws for selected parameters.

*Usage:*

```
MCMC_Fit$draws(
  pars = NULL,
  chains = NULL,
  best_chains = NULL,
  inc_random = FALSE,
  inc_transform = TRUE,
  inc_generate = TRUE
)
```

*Arguments:*

`pars` Character or numeric vector specifying the names or indices of parameters to extract. If NULL, all available parameters are extracted.

`chains` Numeric vector specifying the chains to extract. If NULL, draws from all chains are returned.

`best_chains` Integer; number of best chains to retain based on mean log-posterior (lp).

`inc_random` Logical; whether to include random effects in the output. Default is FALSE.

`inc_transform` Logical; whether to include transformed parameters in the output. Default is TRUE.

`inc_generate` Logical; whether to include generated quantities in the output. Default is TRUE.

*Returns:* Posterior draws.

**Method** `summary()`: Summarize posterior draws.

*Usage:*

```
MCMC_Fit$summary(
  pars = NULL,
  chains = NULL,
  best_chains = NULL,
  max_rows = 10,
  digits = 2,
  inc_random = FALSE,
  inc_transform = TRUE,
  inc_generate = TRUE
)
```

*Arguments:*

`pars` Character or numeric vector specifying the names or indices of parameters to summarize. If NULL, all available parameters are summarized.

`chains` Numeric vector specifying the chains to extract. If NULL, draws from all chains are used.

`best_chains` Integer; number of best chains to retain based on mean log-posterior (lp).

`max_rows` Integer; maximum number of rows to print in the summary table. Default is 10.

`digits` Integer; number of decimal places to print. Default is 2.

`inc_random` Logical; whether to include random effects in the summary. Default is FALSE.

`inc_transform` Logical; whether to include transformed parameters in the summary. Default is TRUE.

`inc_generate` Logical; whether to include generated quantities in the summary. Default is TRUE.

*Returns:* A summary object.

**Method** `rhat_summary()`: Summarize R-hat values.

*Usage:*

```
MCMC_Fit$rhat_summary(
  pars = NULL,
  chains = NULL,
  best_chains = NULL,
  inc_random = FALSE,
  inc_transform = TRUE,
  inc_generate = FALSE,
  finite = TRUE,
  ...
)
```

*Arguments:*

`pars` Character or numeric vector specifying parameters to include.

`chains` Numeric vector specifying chains to include.

`best_chains` Integer; number of best chains to retain based on mean log-posterior.

`inc_random` Logical; whether to include random effects. Default is FALSE.

`inc_transform` Logical; whether to include transformed parameters. Default is TRUE.

`inc_generate` Logical; whether to include generated quantities. Default is FALSE.

`finite` Logical; whether to drop non-finite or missing R-hat values. Default is TRUE.

... Additional arguments.

*Returns:* A numeric vector of R-hat values with class "rhat\_summary".

**Method** `unconstrain_draws()`: Transform posterior draws to the unconstrained scale.

*Usage:*

```
MCMC_Fit$unconstrain_draws()
```

*Returns:* Posterior draws on the unconstrained scale.

**Method** `log_prob()`: Evaluate log-probability values.

*Usage:*

```
MCMC_Fit$log_prob(safe = FALSE)
```

*Arguments:*

`safe` Logical; whether to wrap the evaluation in a tryCatch block. Default is FALSE for speed.

*Returns:* Numeric vector of log-probability values.

**Method** `bridgesampling()`: Estimate the marginal likelihood by bridge sampling.

*Usage:*

```
MCMC_Fit$bridgesampling(
  method = "normal",
  use_neff = TRUE,
  seed = NULL,
  max_iter = 100
)
```

*Arguments:*

*method* Character; the method to use for bridge sampling (e.g., "warp3", "normal"). Default is "warp3".

*use\_neff* Logical; whether to use the effective sample size (ESS) to adjust for autocorrelation. Default is TRUE.

*seed* Integer; random seed for reproducibility. Default is NULL.

*max\_iter* Integer; maximum number of iterations for the estimation algorithm. Default is 100.

*Returns:* Bridge sampling result.

**Method** `WAIC()`: Compute WAIC from pointwise generated log likelihood.

*Usage:*

```
MCMC_Fit$WAIC(...)
```

*Arguments:*

... Additional arguments passed to `'draws()'`, such as `'chains'` or `'best_chains'`.

*Returns:* A `'waic_BayesRTMB'` object.

**Method** `diagnose()`: Run basic diagnostics for the MCMC fit.

*Usage:*

```
MCMC_Fit$diagnose(...)
```

*Arguments:*

... Additional arguments passed to `'diagnose_mcmc_fit()'`.

*Returns:* A `'diagnose_BayesRTMB'` object.

**Method** `bayes_factor()`: Calculate the Bayes factor by marginal-likelihood model comparison.

*Usage:*

```
MCMC_Fit$bayes_factor(
  fixed = NULL,
  comparison_fit = NULL,
  bs_method = "normal",
  error_threshold = 0.2,
  ...
)
```

*Arguments:*

*fixed* Named list of parameter values used to construct the comparison model. For example, `fixed = list(delta = 0)` or `fixed = list("b[x]" = 0)`.

*comparison\_fit* Optional `'MCMC_Fit'` object for an already fitted comparison model.

`bs_method` Character; the method to use for bridge sampling ("normal" or "warp3").  
`error_threshold` Numeric; threshold for the approximate error warning.  
 ... Additional arguments passed to 'sample()' when fitting the comparison model (e.g., `chains = 4`, `sampling = 4000`).

*Returns:* A list of class `bayes_factor_rtmb` containing Bayes factor results.

**Method** `transformed_draws()`: Compute transformed parameters from posterior draws.

*Usage:*

```
MCMC_Fit$transformed_draws(
  tran_fn = NULL,
  progress = c("auto", "none", "bar", "message"),
  tape = c("auto", "none", "force")
)
```

*Arguments:*

`tran_fn` A function for transformed parameters.  
`progress` Progress reporting style: "auto", "none", "bar", or "message". "auto" and "bar" use line-based messages.  
`tape` Derived-quantity evaluation mode. "auto" tries RTMB tape evaluation and falls back to R evaluation on error; "none" always uses R evaluation; "force" requires tape evaluation and errors if it fails.

*Returns:* Transformed parameter draws.

**Method** `generated_quantities()`: Compute generated quantities from posterior draws.

*Usage:*

```
MCMC_Fit$generated_quantities(
  code,
  progress = c("auto", "none", "bar", "message"),
  tape = c("auto", "none", "force")
)
```

*Arguments:*

`code` An 'rtmb\_code({ ... })' or '{ ... }' block containing the logic to be calculated using posterior samples.  
`progress` Progress reporting style: "auto", "none", "bar", or "message". "auto" and "bar" use line-based messages.  
`tape` Derived-quantity evaluation mode. "auto" tries RTMB tape evaluation and falls back to R evaluation on error; "none" always uses R evaluation; "force" requires tape evaluation and errors if it fails.

*Returns:* The 'MCMC\_Fit' object itself (invisibly). Results are appended to the 'generate\_fit' field.

**Method** `resolve_switching()`: Resolve label switching in posterior draws.

*Usage:*

```
MCMC_Fit$resolve_switching(
  target,
  linked = NULL,
  overwrite = TRUE,
  scalar_fns = list()
)
```

*Arguments:*

**target** Character string specifying the target variable to base the relabeling on.

**linked** Character vector of variable names to be relabeled in the same order as the target. Default is NULL.

**overwrite** Logical; whether to overwrite the stored draws in the current object. Default is TRUE.

**scalar\_fns** A named list of functions to apply to scalar variables for relabeling. Default is an empty list.

*Returns:* Relabeled draws or updated object.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
MCMC_Fit$clone(deep = FALSE)
```

*Arguments:*

**deep** Whether to make a deep clone.

---

parameters\_code

*Code block for parameter definitions*

---

## Description

Defines the list of parameters to be passed to ‘rtmb\_model’. By writing in the format ‘variable\_name = Dim(...)’ within a block enclosed in “, evaluation is delayed, enabling strict error checking during model construction.

## Usage

```
parameters_code(expr)
```

## Arguments

**expr** A parameter definition expression enclosed in “.

## Value

A lazily evaluated list of parameters.

**Description**

When declaring parameters in the parameters block using `Dim()`, you can specify a type to automatically apply structural constraints. These constraints ensure that parameters remain within their valid mathematical space during estimation.

**Details****Standard Types:**

- "vector", "matrix", "array": Standard unconstrained containers. Use lower and upper for element-wise bounds.

**Ordering Constraints:**

- "ordered": A vector where  $x_1 < x_2 < \dots < x_K$ .
- "positive\_ordered": A vector where  $0 < x_1 < x_2 < \dots < x_K$ .

**Constrained Vectors:**

- "simplex": A vector where all elements are  $\geq 0$  and  $\text{sum}(x) = 1$ .
- "centered": A vector of length  $K$  where  $\text{sum}(x) = 0$ . (Estimated using  $K-1$  degrees of freedom).

**Correlation and Covariance Matrices:**

- "corr\_matrix": A symmetric, positive-definite correlation matrix (diagonal elements are 1).
- "cov\_matrix": A symmetric, positive-definite covariance matrix.
- "CF\_corr": The Cholesky factor of a correlation matrix.
- "CF\_cov": The Cholesky factor of a covariance matrix (diagonal elements are positive).

**Special Structural Matrices:**

- "lower\_tri": A lower-triangular matrix.
- "positive\_lower\_tri": A lower-triangular matrix with positive diagonal elements.
- "centered\_matrix": A matrix where each column sums to zero.
- "centered\_tri": A triangular matrix with column-wise sum-to-zero constraints (often used for identification in factor analysis).

**See Also**

[rtmb\\_code](#), [math\\_functions](#), [distributions](#)

---

|              |                                               |
|--------------|-----------------------------------------------|
| plot.ce_rtmb | <i>Plot method for ce_rtmb class (Base R)</i> |
|--------------|-----------------------------------------------|

---

**Description**

Plot method for ce\_rtmb class (Base R)

**Usage**

```
## S3 method for class 'ce_rtmb'
plot(x, ...)
```

**Arguments**

|     |                            |
|-----|----------------------------|
| x   | An object of class ce_rtmb |
| ... | Additional arguments.      |

---

|                   |                                            |
|-------------------|--------------------------------------------|
| plot.rtmb_lsmeans | <i>Plot marginal means with error bars</i> |
|-------------------|--------------------------------------------|

---

**Description**

Plot method for rtmb\_lsmeans objects.

**Usage**

```
## S3 method for class 'rtmb_lsmeans'
plot(
  x,
  y,
  error_bar = "se",
  col = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  ...
)
```

**Arguments**

|           |                                   |
|-----------|-----------------------------------|
| x         | An rtmb_lsmeans object.           |
| y         | Ignored.                          |
| error_bar | Type of error bar ("se" or "ci"). |
| col       | Color of the bars.                |

|      |                                         |
|------|-----------------------------------------|
| main | Plot title.                             |
| xlab | X axis label.                           |
| ylab | Y axis label.                           |
| ...  | Additional arguments passed to barplot. |

---

|          |                                                            |
|----------|------------------------------------------------------------|
| plot_acf | <i>Plot autocorrelation for one variable across chains</i> |
|----------|------------------------------------------------------------|

---

**Description**

Draw autocorrelation plots for a selected parameter across all chains.

**Usage**

```
plot_acf(x, var_idx = 1)
```

**Arguments**

|         |                                                                                    |
|---------|------------------------------------------------------------------------------------|
| x       | A 3D array of posterior samples with dimensions ‘(iterations, chains, variables)’. |
| var_idx | Integer. Index of the variable to plot.                                            |

**Value**

No return value. This function is called for its side effect of plotting.

---

|                          |                                 |
|--------------------------|---------------------------------|
| plot_conditional_effects | <i>Plot conditional effects</i> |
|--------------------------|---------------------------------|

---

**Description**

Calculate and plot conditional effects for a fitted model.

**Usage**

```
plot_conditional_effects(
  fit,
  effect,
  prob = 0.95,
  sd_multiplier = 1,
  sd_slice = NULL,
  resolution = 100,
  ...
)
```

**Arguments**

|               |                                                                                                          |
|---------------|----------------------------------------------------------------------------------------------------------|
| fit           | A fitted model object (e.g., 'MCMC_Fit', 'MAP_Fit', 'VB_Fit').                                           |
| effect        | Name of the variable to visualize (e.g., "X1" or "X1:X2").                                               |
| prob          | Probability for the credible/confidence interval.                                                        |
| sd_multiplier | Numeric multiplier for standard deviation when splitting continuous moderators.                          |
| sd_slice      | Logical or NULL; controls whether continuous moderators are evaluated at mean - SD, mean, and mean + SD. |
| resolution    | Grid resolution to calculate for continuous variables.                                                   |
| ...           | Additional arguments passed to the plot method.                                                          |

**Value**

Invisibly returns the plotted object of class `ce_rtmf`.

---

|           |                                                  |
|-----------|--------------------------------------------------|
| plot_dens | <i>Plot posterior densities for MCMC samples</i> |
|-----------|--------------------------------------------------|

---

**Description**

Draw kernel density plots for each parameter across chains.

**Usage**

```
plot_dens(x, mono = FALSE)
```

**Arguments**

|      |                                                                                                                                                                            |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | A 3D array of posterior samples with dimensions '(iterations, chains, variables)'. A 2D matrix '(iterations, chains)' is also allowed and is treated as a single variable. |
| mono | Logical. If 'TRUE', plot in monochrome. If 'FALSE', use blue shades.                                                                                                       |

**Value**

No return value. This function is called for its side effect of plotting.

---

|             |                                                                      |
|-------------|----------------------------------------------------------------------|
| plot_forest | <i>Plot parameter estimates and credible intervals (Forest Plot)</i> |
|-------------|----------------------------------------------------------------------|

---

**Description**

Plot parameter estimates and credible intervals (Forest Plot)

**Usage**

```
plot_forest(
  x,
  prob = 0.95,
  point_estimate = c("median", "EAP", "mean", "MAP", "marginal_map", "joint_map")
)
```

**Arguments**

|                |                                                                                                                                                                                   |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x              | A 3D array of posterior samples with dimensions ‘(iterations, chains, variables)’.                                                                                                |
| prob           | Numeric. Probability mass for the credible interval (default is 0.95).                                                                                                            |
| point_estimate | Character. Point estimate shown as dots. One of “median” (default), “EAP”/“mean”, “MAP”/“marginal_map”, or “joint_map”. “joint_map” uses the draw with the largest ‘lp’ variable. |

**Value**

No return value.

---

|                 |                                           |
|-----------------|-------------------------------------------|
| plot_item_curve | <i>Plot item/category response curves</i> |
|-----------------|-------------------------------------------|

---

**Description**

Calculate and plot item/category response curves for a fitted IRT model.

**Usage**

```
plot_item_curve(fit, ...)
```

**Arguments**

|     |                                                                               |
|-----|-------------------------------------------------------------------------------|
| fit | A fitted IRT model object.                                                    |
| ... | Additional arguments passed to <a href="#">item_curve</a> or the plot method. |

**Value**

Invisibly returns the plotted object of class `rtmb_item_curve`.

---

|                |                                        |
|----------------|----------------------------------------|
| plot_item_info | <i>Plot item information functions</i> |
|----------------|----------------------------------------|

---

**Description**

Calculate and plot item information functions for a fitted IRT model.

**Usage**

```
plot_item_info(fit, ...)
```

**Arguments**

|     |                                                                              |
|-----|------------------------------------------------------------------------------|
| fit | A fitted IRT model object.                                                   |
| ... | Additional arguments passed to <a href="#">item_info</a> or the plot method. |

**Value**

Invisibly returns the plotted object of class `rtmb_item_info`.

---

|              |                                          |
|--------------|------------------------------------------|
| plot_lsmeans | <i>Plot least-squares marginal means</i> |
|--------------|------------------------------------------|

---

**Description**

Calculate and plot marginal means for a fitted model.

**Usage**

```
plot_lsmeans(fit, specs, ...)
```

**Arguments**

|       |                                                                                    |
|-------|------------------------------------------------------------------------------------|
| fit   | A fitted model object (e.g. 'Classic_Fit').                                        |
| specs | Character vector specifying the variables for which marginal means are calculated. |
| ...   | Additional arguments passed to <a href="#">lsmeans</a> or the plot method.         |

**Value**

Invisibly returns the plotted object.

plot\_mdu

*Plot Multidimensional Unfolding Configuration***Description**

Draws an item-person map for multidimensional unfolding models. Item locations are shown as labels, optional blue circles show item alpha/radius values, and the respondent locations are summarized by a two- dimensional kernel density contour.

**Usage**

```
plot_mdu(
  delta,
  theta = NULL,
  item_alpha = NULL,
  phi = NULL,
  dims = c(1, 2),
  radius = NULL,
  signs = c(1, 1),
  item_labels = NULL,
  show_radius = TRUE,
  show_density = TRUE,
  circle_scale = 1,
  alpha = 0.2,
  contour_n = 60,
  distance = c("auto", "euclidean", "squared"),
  point_estimate = c("EAP", "MAP", "mean", "marginal_map", "joint_map"),
  prefer_rotated = TRUE,
  show_phi = NULL,
  main = "MDU Configuration",
  xlab = NULL,
  ylab = NULL,
  ...
)
```

**Arguments**

|            |                                                                                                                                                |
|------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| delta      | Item coordinate matrix (M items x D dimensions), or a fitted object with an 'EAP()' method.                                                    |
| theta      | Person coordinate matrix (N persons x D dimensions). If 'delta' is a fitted object and 'theta' is 'NULL', it is extracted from the fit.        |
| item_alpha | Optional item alpha vector used for item radii. If 'delta' is a fitted object and 'item_alpha' is 'NULL', 'alpha' is extracted when available. |
| phi        | Deprecated alias for 'item_alpha'.                                                                                                             |
| dims       | Integer vector of length 2 specifying dimensions to plot.                                                                                      |

|                  |                                                                                                                                                                                                     |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| radius           | Numeric plot radius. If 'NULL', a radius is chosen from the coordinates.                                                                                                                            |
| signs            | Numeric vector of length 2. Use '-1' to flip an axis.                                                                                                                                               |
| item_labels      | Optional item labels. Defaults to row names or item numbers.                                                                                                                                        |
| show_radius      | Logical; whether to draw radius circles based on item alpha.                                                                                                                                        |
| show_density     | Logical; whether to draw density contours for 'theta'.                                                                                                                                              |
| circle_scale     | Numeric multiplier for item radii.                                                                                                                                                                  |
| alpha            | Circle transparency.                                                                                                                                                                                |
| contour_n        | Grid size passed to 'MASS::kde2d()'.                                                                                                                                                                |
| distance         | Character; "auto", "euclidean", or "squared". Used to transform item alpha values into plotted radii. "auto" uses the fit's stored distance when available and otherwise falls back to "euclidean". |
| point_estimate   | Character; point estimate used when 'delta' is a fit object. Passed to 'estimate()'.                                                                                                                |
| prefer_rotated   | Logical; when 'delta' is a fitted object, prefer rotated generated quantities ('delta_rot' and 'theta_rot') if available.                                                                           |
| show_phi         | Deprecated alias for 'show_radius'.                                                                                                                                                                 |
| main, xlab, ylab | Plot title and axis labels.                                                                                                                                                                         |
| ...              | Additional arguments passed to 'plot()'.                                                                                                                                                            |

**Value**

Invisibly returns a list with plotted 'delta', 'theta', and 'item\_alpha'.

---

|            |                                         |
|------------|-----------------------------------------|
| plot_pairs | <i>Plot pairs for posterior samples</i> |
|------------|-----------------------------------------|

---

**Description**

Draw a scatterplot matrix to examine posterior correlations between parameters.

**Usage**

```
plot_pairs(x, pars = NULL)
```

**Arguments**

|      |                                                                                                                                                               |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | A 3D array of posterior samples with dimensions '(iterations, chains, variables)'.                                                                            |
| pars | Character vector or integer vector specifying which parameters to plot. If NULL (default), up to the first 10 parameters are plotted to prevent overplotting. |

**Value**

No return value.

---

|                |                                       |
|----------------|---------------------------------------|
| plot_test_info | <i>Plot test information function</i> |
|----------------|---------------------------------------|

---

**Description**

Calculate and plot test information function for a fitted IRT model.

**Usage**

```
plot_test_info(fit, ...)
```

**Arguments**

|     |                                                                           |
|-----|---------------------------------------------------------------------------|
| fit | A fitted IRT model object.                                                |
| ... | Additional arguments passed to <code>test_info</code> or the plot method. |

**Value**

Invisibly returns the plotted object of class `rtmb_test_info`.

---

|            |                              |
|------------|------------------------------|
| plot_trace | <i>Plot MCMC trace plots</i> |
|------------|------------------------------|

---

**Description**

Draw trace plots for each parameter across chains.

**Usage**

```
plot_trace(x, mono = FALSE)
```

**Arguments**

|      |                                                                                                                                                                            |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x    | A 3D array of posterior samples with dimensions ‘(iterations, chains, variables)’. A 2D matrix ‘(iterations, chains)’ is also allowed and is treated as a single variable. |
| mono | Logical. If ‘TRUE’, plot in monochrome. If ‘FALSE’, use blue shades.                                                                                                       |

**Value**

No return value. This function is called for its side effect of plotting.

---

|                   |                                        |
|-------------------|----------------------------------------|
| posterior_predict | <i>Posterior predictive simulation</i> |
|-------------------|----------------------------------------|

---

**Description**

Generate replicated outcomes from an estimated BayesRTMB model. Regression wrappers provide an automatic simulator. For a custom model, supply an `'rtmb_code()'` object containing a `'generate'` block that creates and reports one replicated outcome vector. MCMC and variational fits use posterior draws. A MAP fit uses sampling-based uncertainty when available and otherwise conditions on its point estimate.

**Usage**

```
posterior_predict(object, ...)

## S3 method for class 'RTMB_Fit_Base'
posterior_predict(object, ...)
```

**Arguments**

|                     |                                                                                 |
|---------------------|---------------------------------------------------------------------------------|
| <code>object</code> | A BayesRTMB fit object.                                                         |
| <code>...</code>    | Arguments passed to the fit object's <code>'posterior_predict()'</code> method. |

**Value**

A numeric matrix with posterior predictive draws in rows and observations in columns.

---

|          |                                    |
|----------|------------------------------------|
| pp_check | <i>Posterior predictive checks</i> |
|----------|------------------------------------|

---

**Description**

Compare observed data with replicated outcomes from an estimated BayesRTMB model. Density overlays are used for continuous outcomes and binned bar plots for discrete outcomes. When `'type = "auto"'`, the display is selected from the likelihood's `'_lpdf'` or `'_lpmf'` implementation.

**Usage**

```
pp_check(object, ...)

## S3 method for class 'RTMB_Fit_Base'
pp_check(object, ...)

## S3 method for class 'rtmb_pp_check'
plot(
```

```

x,
main = NULL,
xlab = NULL,
ylab = NULL,
observed_col = "#1B1B1B",
predictive_col = "#2C7FB8",
show_legend = TRUE,
legend_position = c("auto", "topleft", "topright", "bottomleft", "bottomright"),
legend_cex = 0.78,
interval = 0.95,
...
)

```

### Arguments

|                 |                                                                                                                           |
|-----------------|---------------------------------------------------------------------------------------------------------------------------|
| object          | A BayesRTMB fit object.                                                                                                   |
| ...             | Arguments passed to the fit object's 'pp_check()' method or to the plotting method.                                       |
| x               | An 'rtmb_pp_check' object.                                                                                                |
| main            | Optional plot title.                                                                                                      |
| xlab            | Optional x-axis label.                                                                                                    |
| ylab            | Optional y-axis label.                                                                                                    |
| observed_col    | Color for observed data.                                                                                                  |
| predictive_col  | Color for predictive data and intervals.                                                                                  |
| show_legend     | Logical; display the plot legend.                                                                                         |
| legend_position | Legend position. "auto" moves scatter-plot legends away from the fitted trend; the other values are passed to 'legend()'. |
| legend_cex      | Relative text and symbol size for the legend.                                                                             |
| interval        | Probability covered by posterior predictive intervals.                                                                    |

### Details

Supplying 'x' to the fit object's 'pp\_check()' method switches to a scatter-based check. Observed outcomes are compared with posterior predictive means and 95 value 'x = ".fitted"' uses the posterior predictive mean on the horizontal axis for a model-wide calibration check.

### Value

An object of class 'rtmb\_pp\_check', returned invisibly after plotting.

---

|                    |                                              |
|--------------------|----------------------------------------------|
| print.bayes_factor | <i>Print method for bayes_factor objects</i> |
|--------------------|----------------------------------------------|

---

**Description**

Print method for bayes\_factor objects

**Usage**

```
## S3 method for class 'bayes_factor'
print(x, digits = 4, ...)
```

**Arguments**

|        |                                                     |
|--------|-----------------------------------------------------|
| x      | An object of class bayes_factor.                    |
| digits | Number of decimal places to display (default is 4). |
| ...    | Additional arguments.                               |

---

|                         |                                                   |
|-------------------------|---------------------------------------------------|
| print.bayes_factor_rtmb | <i>Print method for bayes_factor_rtmb objects</i> |
|-------------------------|---------------------------------------------------|

---

**Description**

Print method for bayes\_factor\_rtmb objects

**Usage**

```
## S3 method for class 'bayes_factor_rtmb'
print(x, digits = 4, ...)
```

**Arguments**

|        |                                                     |
|--------|-----------------------------------------------------|
| x      | An object of class bayes_factor_rtmb.               |
| digits | Number of decimal places to display (default is 4). |
| ...    | Additional arguments.                               |

---

|               |                                                                  |
|---------------|------------------------------------------------------------------|
| print.ce_rtmb | <i>Print method for ce_rtmb class (automatically calls plot)</i> |
|---------------|------------------------------------------------------------------|

---

**Description**

Print method for ce\_rtmb class (automatically calls plot)

**Usage**

```
## S3 method for class 'ce_rtmb'  
print(x, ...)
```

**Arguments**

|     |                            |
|-----|----------------------------|
| x   | An object of class ce_rtmb |
| ... | Additional arguments.      |

---

|                 |                             |
|-----------------|-----------------------------|
| print.ce_simple | <i>Print simple effects</i> |
|-----------------|-----------------------------|

---

**Description**

Print simple effects

**Usage**

```
## S3 method for class 'ce_simple'  
print(x, digits = 3, ...)
```

**Arguments**

|        |                            |
|--------|----------------------------|
| x      | A 'ce_simple' object.      |
| digits | Number of digits to print. |
| ...    | Additional arguments.      |

---

```
print.summary_BayesRTMB
      print for summary_BayesRTMB class
```

---

### Description

print for summary\_BayesRTMB class

### Usage

```
## S3 method for class 'summary_BayesRTMB'
print(x, digits = NULL, ...)
```

### Arguments

|        |                                      |
|--------|--------------------------------------|
| x      | An object of class summary_BayesRTMB |
| digits | integer                              |
| ...    | Additional arguments.                |

---

|            |                             |
|------------|-----------------------------|
| prior_flat | <i>Specify a flat prior</i> |
|------------|-----------------------------|

---

### Description

‘prior\_flat()’ specifies that no additional prior density is added by the wrapper. This is the only prior type allowed for ‘classic()’.

### Usage

```
prior_flat()
```

### Value

A list with class “rtmb\_prior” and ‘type = “flat”’.

---

|           |                                                               |
|-----------|---------------------------------------------------------------|
| prior_jzs | <i>Specify a JZS (Jeffrey-Zellner-Siow) prior for t-tests</i> |
|-----------|---------------------------------------------------------------|

---

**Description**

Specify a JZS (Jeffrey-Zellner-Siow) prior for t-tests

**Usage**

```
prior_jzs(r = 0.707)
```

**Arguments**

|   |                                                                                       |
|---|---------------------------------------------------------------------------------------|
| r | Numeric; scale parameter for the Cauchy prior on effect size delta. Default is 0.707. |
|---|---------------------------------------------------------------------------------------|

**Value**

A list with class "rtmb\_prior"

---

|              |                                                                         |
|--------------|-------------------------------------------------------------------------|
| prior_normal | <i>Specify normal/exponential priors for MAP and Bayesian inference</i> |
|--------------|-------------------------------------------------------------------------|

---

**Description**

'prior\_normal()' specifies normal priors for location parameters and exponential priors for scale parameters. It is intended for MAP and Bayesian inference, not for 'classic()'.

**Usage**

```
prior_normal(
  Intercept_sd = 10,
  mu_sd = 10,
  b_sd = 10,
  sigma_rate = 1/5,
  tau_rate = 1/5,
  ...
)
```

**Arguments**

|              |                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| Intercept_sd | Standard deviation for the intercept prior. If 'NULL', no intercept prior is added.                                                             |
| mu_sd        | Standard deviation for mean/intercept priors. If 'NULL', no mean prior is added.                                                                |
| b_sd         | Standard deviation for coefficient priors. If 'NULL', no coefficient prior is added.                                                            |
| sigma_rate   | Rate for residual standard deviation priors. The default '1 / 5' gives an exponential prior with mean 5. If 'NULL', no sigma prior is added.    |
| tau_rate     | Rate for random-effect standard deviation priors. The default '1 / 5' gives an exponential prior with mean 5. If 'NULL', no tau prior is added. |
| ...          | Optional wrapper-specific hyperparameters.                                                                                                      |

**Value**

A list with class "rtmb\_prior" and 'type = "normal"'.

---

prior\_rhs

*Specify a Regularized Horseshoe prior for continuous shrinkage*

---

**Description**

Specify a Regularized Horseshoe prior for continuous shrinkage

**Usage**

```
prior_rhs(expected_vars = 3, slab_scale = 2, slab_df = 4, ...)
```

**Arguments**

|               |                                                               |
|---------------|---------------------------------------------------------------|
| expected_vars | Expected number of non-zero variables. Default is 3.          |
| slab_scale    | Scale parameter for the slab distribution. Default is 2.0.    |
| slab_df       | Degrees of freedom for the slab distribution. Default is 4.0. |
| ...           | Optional hyperparameters for other distributions.             |

**Value**

A list with class "rtmb\_prior"

---

|           |                                                              |
|-----------|--------------------------------------------------------------|
| prior_ssp | <i>Specify a Spike-and-Slab prior for variable selection</i> |
|-----------|--------------------------------------------------------------|

---

### Description

Specify a Spike-and-Slab prior for variable selection

### Usage

```
prior_ssp(ssp_ratio = 0.25, max_beta = 1, ...)
```

### Arguments

|           |                                                                    |
|-----------|--------------------------------------------------------------------|
| ssp_ratio | Prior probability of inclusion for each variable. Default is 0.25. |
| max_beta  | Maximum expected effect size. Default is 1.0.                      |
| ...       | Optional hyperparameters for other distributions.                  |

### Value

A list with class "rtmb\_prior"

---

|               |                             |
|---------------|-----------------------------|
| prior_uniform | <i>Specify a flat prior</i> |
|---------------|-----------------------------|

---

### Description

Deprecated alias of 'prior\_flat()'.

### Usage

```
prior_uniform()
```

### Value

A list with class "'rtmb\_prior'" and 'type = "flat"'.

---

|            |                                           |
|------------|-------------------------------------------|
| prior_weak | <i>Specify a weakly informative prior</i> |
|------------|-------------------------------------------|

---

**Description**

Specify a weakly informative prior

**Usage**

```
prior_weak(sd_ratio = 0.5, max_beta = 1, ...)
```

**Arguments**

|          |                                                                                                   |
|----------|---------------------------------------------------------------------------------------------------|
| sd_ratio | Ratio of the prior standard deviation to the half-range of the response variable. Default is 0.5. |
| max_beta | Maximum expected effect size. Default is 1.0.                                                     |
| ...      | Optional hyperparameters for other distributions.                                                 |

**Value**

A list with class "rtmb\_prior"

---

|                |                                               |
|----------------|-----------------------------------------------|
| quad_form_chol | <i>Quadratic form using a Cholesky factor</i> |
|----------------|-----------------------------------------------|

---

**Description**

Quadratic form using a Cholesky factor

**Usage**

```
quad_form_chol(x, L)
```

**Arguments**

|   |                                            |
|---|--------------------------------------------|
| x | A numeric vector.                          |
| L | A lower triangular Cholesky factor matrix. |

**Value**

The value of the quadratic form.

---

|                |                                              |
|----------------|----------------------------------------------|
| quad_form_diag | <i>Quadratic form with a diagonal matrix</i> |
|----------------|----------------------------------------------|

---

**Description**

Quadratic form with a diagonal matrix

**Usage**

```
quad_form_diag(m, v)
```

**Arguments**

|   |                                                      |
|---|------------------------------------------------------|
| m | A numeric matrix.                                    |
| v | A numeric vector representing the diagonal elements. |

**Value**

The value of the quadratic form.

---

|            |                                |
|------------|--------------------------------|
| quantile95 | <i>Calculate 95% Quantiles</i> |
|------------|--------------------------------|

---

**Description**

Calculate 95% Quantiles

**Usage**

```
quantile95(x)
```

**Arguments**

|   |                   |
|---|-------------------|
| x | A numeric vector. |
|---|-------------------|

**Value**

A numeric vector of length 2.

---

|               |                                  |
|---------------|----------------------------------|
| read_mcmc_csv | <i>Restore MCMC Fit from CSV</i> |
|---------------|----------------------------------|

---

**Description**

Restore MCMC Fit from CSV

**Usage**

```
read_mcmc_csv(model, name, dir = "BayesRTMB_mcmc", chains = 4, laplace = FALSE)
```

**Arguments**

|         |                                                                    |
|---------|--------------------------------------------------------------------|
| model   | An RTMB_Model object.                                              |
| name    | Base name of the saved CSVs.                                       |
| dir     | Directory where CSVs are saved. Default is "BayesRTMB_mcmc".       |
| chains  | Number of chains. Default is 4.                                    |
| laplace | Logical; whether Laplace approximation was used. Default is FALSE. |

**Value**

An MCMC\_Fit object.

---

|              |                                    |
|--------------|------------------------------------|
| rhat_summary | <i>Summarize MCMC R-hat Values</i> |
|--------------|------------------------------------|

---

**Description**

Extract R-hat values from a fitted MCMC object as a numeric vector. The returned object prints a compact convergence summary and can be passed directly to functions such as [hist](#).

**Usage**

```
rhat_summary(fit, ...)

## Default S3 method:
rhat_summary(fit, ...)

## S3 method for class 'mcmc_fit'
rhat_summary(
  fit,
  pars = NULL,
  chains = NULL,
  best_chains = NULL,
```

```

    inc_random = FALSE,
    inc_transform = TRUE,
    inc_generate = FALSE,
    finite = TRUE,
    ...
  )

```

### Arguments

|                            |                                                                                                                 |
|----------------------------|-----------------------------------------------------------------------------------------------------------------|
| <code>fit</code>           | A fitted model object.                                                                                          |
| <code>...</code>           | Additional arguments passed to methods.                                                                         |
| <code>pars</code>          | Character or numeric vector specifying parameters to include. If NULL, all available parameters are summarized. |
| <code>chains</code>        | Numeric vector specifying chains to include.                                                                    |
| <code>best_chains</code>   | Integer; number of best chains to retain based on mean log-posterior.                                           |
| <code>inc_random</code>    | Logical; whether to include random effects. Default is FALSE.                                                   |
| <code>inc_transform</code> | Logical; whether to include transformed parameters. Default is TRUE.                                            |
| <code>inc_generate</code>  | Logical; whether to include generated quantities. Default is FALSE.                                             |
| <code>finite</code>        | Logical; whether to drop non-finite or missing R-hat values. Default is TRUE.                                   |

### Value

A numeric vector of R-hat values with class "rhat\_summary".

### Examples

```

data(debate, package = "BayesRTMB")
mdl <- rtmb_lm(sat ~ talk + perf, data = debate)
fit <- mdl$sample(sampling = 200, warmup = 200, chains = 2)
r <- rhat_summary(fit)
r
hist(r)

```

---

|            |                               |
|------------|-------------------------------|
| rtmb_array | Create an AD-compatible array |
|------------|-------------------------------|

---

### Description

Creates a fixed-size vector, matrix, or higher-dimensional array that is safe to use inside 'rtmb\_code()' when subsequent assignments may involve RTMB automatic-differentiation values. This is a safer alternative to 'array(value, dim)' or 'matrix(value, ...)' in model code.

### Usage

```
rtmb_array(value = 0, dim, seed = NULL)
```

**Arguments**

|       |                                                                      |
|-------|----------------------------------------------------------------------|
| value | Initial value. Usually a scalar such as '0'.                         |
| dim   | Integer dimension vector.                                            |
| seed  | Optional AD value used to create the container. See 'rtmb_vector()'. |

**Value**

An AD-compatible array.

---

rtmb\_code

*Define an RTMB Model with Stan-like Syntax*


---

**Description**

rtmb\_code is the primary interface for defining Bayesian or Frequentist models within the RTMB framework. It allows you to organize model logic into functional blocks while utilizing a Stan-like ~ operator for specifying priors and likelihoods.

**Usage**

```
rtmb_code(...)
```

**Arguments**

...                    Model definition blocks.

**Details****Key Blocks and Logic:**

- setup: Pre-compilation data processing.
- parameters: Declaration of estimated parameters. See [parameter\\_types](#) for available constraints.
- transform: Definition of intermediate variables. Use [math\\_functions](#) for stable AD calculations.
- model: Likelihood and Priors. Refer to [distributions](#) for available sampling statements.
- generate: Post-hoc calculation of predictions or diagnostics.

**Automatic Differentiation (AD) Note:** All code within parameters, transform, and model is automatically differentiated by RTMB. To ensure numerical stability, use provided utility functions like [log\\_sum\\_exp](#) or [inv\\_logit](#) instead of raw algebraic implementations.

**Value**

A list of unevaluated code blocks.

**See Also**

[rtmb\\_model](#) for building the model object. [parameter\\_types](#) for constraining parameters. [distributions](#) for likelihood functions. [math\\_functions](#) for numerical utilities.

**Examples**

```
# Simulate data for a linear regression
set.seed(123)
N <- 100
x <- rnorm(N)
y <- 2.0 + 1.5 * x + rnorm(N, mean = 0, sd = 1)
data_list <- list(N = N, x = x, y = y)

# Define the model using rtmb_code
code <- rtmb_code(
  setup = {
    # Center the predictor variable (executed once)
    x_centered <- x - mean(x)
  },
  parameters = {
    # Define parameters and their constraints
    alpha = Dim(1)
    beta = Dim(1)
    sigma = Dim(1, lower = 0)
  },
  transform = {
    # Calculate the linear predictor
    mu <- alpha + beta * x_centered
  },
  model = {
    # Priors
    alpha ~ normal(0, 10)
    beta ~ normal(0, 10)
    sigma ~ exponential(1)

    # Likelihood (Vectorized)
    y ~ normal(mu, sigma)
  },
  generate = {
    # Calculate generated quantities
    y_pred <- mu

    # Must return a named list
    list(y_pred = y_pred)
  }
)

# Create the model object
mod <- rtmb_model(data = data_list, code = code)

# Fit the model using MAP estimation
map_res <- mod$optimize()
```

```
map_res$summary(pars = c("alpha", "beta", "sigma"))

# The generated quantity 'y_pred' can also be summarized
map_res$summary("y_pred", max_rows = 5)
```

---

rtmb\_corr

*Fit a Correlation Model using RTMB*


---

## Description

‘rtmb\_corr’ fits a correlation model to estimate means, standard deviations, and correlation structures. It supports simple correlation, multilevel correlation, and classical frequentist estimation.

## Usage

```
rtmb_corr(
  x = NULL,
  data = NULL,
  ID = NULL,
  covariates = NULL,
  method = c("pearson", "spearman", "reml"),
  prior = prior_flat(),
  y_range = NULL,
  init = NULL,
  fixed = NULL,
  missing = c("listwise", "fiml", "pairwise"),
  WAIC = FALSE
)
```

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                                             |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x          | A matrix, data frame, formula, or expression (e.g., <code>cbind(V1, V2)</code> ) of response variables.                                                                                                                                                                                                                                                     |
| data       | An optional data frame containing the variables.                                                                                                                                                                                                                                                                                                            |
| ID         | A character string or expression specifying the group ID variable for multilevel models.                                                                                                                                                                                                                                                                    |
| covariates | Optional numeric matrix or data frame of covariates to be included in the joint MVN model.                                                                                                                                                                                                                                                                  |
| method     | Correlation method for <code>classic()</code> : "pearson", "spearman", or "reml".                                                                                                                                                                                                                                                                           |
| prior      | Prior configuration object: ‘prior_flat()’, ‘prior_normal()’, or ‘prior_weak()’. Default is ‘prior_flat()’. For ‘prior_normal()’, correlation models use the aliases ‘mean_sd’ and ‘sd_rate’, which override the common arguments ‘mu_sd’ and ‘sigma_rate’. An LKJ prior with ‘lkj_eta = 1’ is added automatically unless ‘lkj_eta’ is supplied explicitly. |

|         |                                                                                                                                                                                                                                         |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| y_range | Optional numeric vector or matrix defining the theoretical range (min, max) of response variables. Required when using <code>prior_weak()</code> . Can be a vector of length 2 (applies to all variables) or a matrix/list of length P. |
| init    | Optional list of initial values.                                                                                                                                                                                                        |
| fixed   | Optional named list of fixed values for specific parameters.                                                                                                                                                                            |
| missing | Missing value handling strategy: "listwise" (default), "pairwise", or "fiml" (Full Information Maximum Likelihood).                                                                                                                     |
| WAIC    | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                               |

### Examples

```
# Estimate the correlation between two variables in the debate dataset
data(debate, package = "BayesRTMB")

fit_corr <- rtmb_corr(cbind(sat, perf), data = debate)

mcmc_corr <- fit_corr$sample(sampling = 500, warmup = 500, chains = 2)
mcmc_corr$summary()

bf_corr <- mcmc_corr$bayes_factor(fixed = list(corr = 0))
print(bf_corr)
```

---

rtmb\_fa

*RTMB-based Factor Analysis Wrapper*


---

### Description

Performs exploratory factor analysis (EFA) using RTMB. It supports standard rotation methods (e.g., varimax, promax) as well as regularized factor analysis using a Spike-and-Slab Prior (SSP) for estimating sparse loading matrices.

### Usage

```
rtmb_fa(
  data,
  nfactors = 1,
  rotate = NULL,
  score = FALSE,
  prior = prior_flat(),
  y_range = NULL,
  init = NULL,
  fixed = NULL,
  missing = c("listwise", "fiml"),
  WAIC = FALSE
)
```

**Arguments**

|                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>     | Observation data frame or matrix (N x J).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>nfactors</code> | Number of factors (K).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>rotate</code>   | String specifying the rotation method (e.g., "varimax", "promax", "ssp"). If NULL, no rotation is applied. Specifying "ssp" performs regularized factor analysis.                                                                                                                                                                                                                                                                                                                                                                               |
| <code>score</code>    | Logical; if TRUE, factor scores are calculated in the generate block (default is FALSE).                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <code>prior</code>    | Prior configuration: 'prior_flat()', 'prior_normal()', or 'prior_weak()'. 'prior_ssp()' is accepted when 'rotate = "ssp"'. Hyperparameters can be specified within these functions (e.g., 'prior_normal(mean_sd = 10, sd_rate = 1 / 5, loadings_sd = 10)'). Available FA aliases are 'mean_sd', 'sd_rate', and 'loadings_sd'; they override the corresponding common arguments 'mu_sd', 'sigma_rate', and 'b_sd'. 'sd_rate' is an exponential rate, so '1 / 5' gives a prior mean of 5. 'ssp_ratio' is available when 'rotate = "ssp"'.<br><br> |
| <code>y_range</code>  | A numeric vector of length 2 specifying the theoretical min and max values of the items. Used to construct weakly informative priors when 'prior = prior_weak()'.                                                                                                                                                                                                                                                                                                                                                                               |
| <code>init</code>     | List of initial values.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <code>fixed</code>    | A named list of parameter values to fix (optional).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>missing</code>  | Missing value handling strategy: "listwise" (default) or "fiml" (Full Information Maximum Likelihood).                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>WAIC</code>     | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Examples**

```
# Prepare a subset of the BigFive dataset for factor analysis
data(BigFive, package = "BayesRTMB")
fa_data <- BigFive[, 1:10]

# --- 1. Standard Exploratory Factor Analysis (1 Factor) ---
fit_fa1 <- rtmb_fa(data = fa_data, nfactors = 1)

# Maximum A Posteriori (MAP) estimation
map_fa1 <- fit_fa1$optimize()
map_fa1$summary()
```

---

RTMB\_Fit\_Base

*Base class for RTMB Fit objects*


---

**Description**

Base class for RTMB Fit objects

Base class for RTMB Fit objects

## Details

An R6 base class providing common methods for Bayesian and MAP inference results.

## Public fields

`model` The 'RTMB\_Model' object used for estimation.

## Methods

### Public methods:

- `RTMB_Fit_Base$get_point_estimate()`
- `RTMB_Fit_Base$estimate()`
- `RTMB_Fit_Base$EAP()`
- `RTMB_Fit_Base$MAP()`
- `RTMB_Fit_Base$rotate()`
- `RTMB_Fit_Base$posterior_predict()`
- `RTMB_Fit_Base$pp_check()`
- `RTMB_Fit_Base$fa_rotate()`
- `RTMB_Fit_Base$clone()`

**Method** `get_point_estimate()`: Abstract method to get a point estimate for a target parameter.

*Usage:*

```
RTMB_Fit_Base$get_point_estimate(target, ...)
```

*Arguments:*

`target` Character string specifying the target parameter name.

`...` Additional arguments.

*Returns:* A numeric array or matrix of the point estimate.

**Method** `estimate()`: Get point estimates for parameters, transformed parameters, and generated quantities.

*Usage:*

```
RTMB_Fit_Base$estimate(
  pars = NULL,
  type = c("mean", "EAP", "marginal_map", "joint_map", "MAP"),
  component = c("all", "parameters", "transform", "generate"),
  chains = NULL,
  best_chains = NULL,
  drop = TRUE,
  ...
)
```

*Arguments:*

`pars` Optional character or numeric vector of parameter names or indices to extract. Supports special keywords: "parameters", "transform", "generate", and "all".

`type` Character string specifying the estimation type.

component Character string specifying the component to filter by.  
 chains Numeric vector of chains to include.  
 best\_chains Number of best chains to include.  
 drop Logical; if TRUE and only one parameter is selected, return the value directly instead of a list.  
 ... Additional arguments passed to draws().

*Returns:* A named list of point estimates, or a single value if 'drop = TRUE'.

**Method EAP():** Calculate Expected A Posteriori (EAP) estimates from posterior samples.

*Usage:*

```
RTMB_Fit_Base$EAP(
  pars = "parameters",
  chains = NULL,
  best_chains = NULL,
  drop = TRUE,
  ...
)
```

*Arguments:*

pars Optional character vector of parameter names to extract.  
 chains Numeric vector of chains to include.  
 best\_chains Number of best chains to include.  
 drop Logical; whether to drop the list if only one parameter is selected.  
 ... Additional arguments passed to 'estimate()'.

*Returns:* A named list of EAP estimates.

**Method MAP():** Calculate Maximum A Posteriori (MAP) estimates.

*Usage:*

```
RTMB_Fit_Base$MAP(
  pars = "parameters",
  chains = NULL,
  best_chains = NULL,
  type = c("marginal", "joint"),
  drop = TRUE,
  ...
)
```

*Arguments:*

pars Optional character vector of parameter names to extract.  
 chains Numeric vector of chains to include.  
 best\_chains Number of best chains to include.  
 type Character string; "marginal" or "joint" MAP.  
 drop Logical; whether to drop the list if only one parameter is selected.  
 ... Additional arguments passed to 'estimate()'.

*Returns:* A named list of MAP estimates.

**Method** rotate(): Rotate sampled parameters.

*Usage:*

```
RTMB_Fit_Base$rotate(
  target,
  reference = NULL,
  linked = NULL,
  principal = FALSE,
  suffix = "rot",
  overwrite = TRUE
)
```

*Arguments:*

target Character string specifying the target variable to base the rotation on.

reference Matrix to rotate towards. If NULL, the target's point estimate is used.

linked Character vector of variable names to be rotated in the same direction.

principal Logical; if TRUE and 'reference' is NULL, the point estimate is first rotated to its principal axes and then used as the Procrustes reference. This is useful for MDU configurations.

suffix Character suffix used for generated quantities. The default stores results as '<variable>\_rot'.

overwrite Logical; if TRUE, existing generated quantities with the same names are overwritten. If FALSE, a numeric suffix is appended.

*Returns:* The updated object invisibly.

**Method** posterior\_predict(): Generate posterior predictive replications.

*Usage:*

```
RTMB_Fit_Base$posterior_predict(
  code = NULL,
  variable = NULL,
  draws = 100L,
  seed = NULL,
  random = c("conditional", "population", "simulate"),
  chains = NULL,
  best_chains = NULL,
  observed = NULL
)
```

*Arguments:*

code Optional 'rtmb\_code()' object containing a 'generate' block for custom models. The block should create and report one replicated outcome.

variable Name of the replicated quantity returned by 'code'. If omitted, 'y\_rep' or the only returned quantity is used.

draws Number of predictive draws, or 'NULL' to use every available posterior draw.

seed Optional random seed.

random Hierarchical prediction mode: "conditional" uses fitted random effects, "population" omits them, and "simulate" draws new random effects for the observed groups.

chains Optional chains to use.

**best\_chains** Optional number of best chains to use.  
**observed** Optional observed outcome or its name in model data. This is stored as metadata when its length matches a prediction.

*Returns:* A matrix with predictive draws in rows and observations in columns.

**Method** `pp_check()`: Plot a posterior predictive check.

*Usage:*

```
RTMB_Fit_Base$pp_check(
  type = c("auto", "dens", "bars"),
  stat = NULL,
  x = NULL,
  code = NULL,
  observed = NULL,
  variable = NULL,
  draws = 100L,
  seed = NULL,
  random = c("conditional", "population", "simulate"),
  chains = NULL,
  best_chains = NULL,
  plot = TRUE,
  ...
)
```

*Arguments:*

**type** Plot type: "dens", "bars", or "auto". Automatic mode uses the observation likelihood's `_lpdf` or `_lpmf` implementation.  
**stat** Optional scalar statistic function, or its name. When supplied, the replicated statistic distribution is compared with its observed value.  
**x** Optional predictor used for a scatter-based check. Supply a column name from the model data or a vector with one value per observation.  
**code** Optional custom generate block; see `'posterior_predict()'`.  
**observed** Optional observed outcome or its name in model data.  
**variable** Name of the replicated quantity returned by `'code'`.  
**draws** Number of posterior predictive replications.  
**seed** Optional random seed.  
**random** Hierarchical prediction mode; see `'posterior_predict()'`.  
**chains** Optional chains to use.  
**best\_chains** Optional number of best chains to use.  
**plot** Logical; draw the check immediately.  
**...** Graphical arguments passed to the plotting method.

*Returns:* An `'rtmb_pp_check'` object, invisibly.

**Method** `fa_rotate()`: Rotate factor loadings and optional factor scores.

*Usage:*

```
RTMB_Fit_Base$fa_rotate(
  target = "L",
  linked = NULL,
  scores = NULL,
  rotate = "promax",
  ...,
  suffix = NULL,
  overwrite = TRUE
)
```

*Arguments:*

**target** Character string specifying the target variable to base the rotation on.  
**linked** Character vector of variable names to be rotated in the same direction.  
**scores** Character vector of variable names to be rotated as factor scores (inverse direction).  
**rotate** Character string specifying the rotation method.  
**...** Additional arguments passed to the rotation function.  
**suffix** Character suffix used for generated quantities. If NULL, the rotation method name is used.  
**overwrite** Logical; if TRUE, existing generated quantities with the same names are overwritten. If FALSE, a numeric suffix is appended.

*Returns:* The updated object invisibly.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
RTMB_Fit_Base$clone(deep = FALSE)
```

*Arguments:*

**deep** Whether to make a deep clone.

---

rtmb\_glm

---

*RTMB-based GLM wrapper function (no random effects)*


---

## Description

RTMB-based GLM wrapper function (no random effects)

## Usage

```
rtmb_glm(
  formula,
  data = NULL,
  family = "gaussian",
  prior = prior_flat(),
  y_range = NULL,
  init = NULL,
  fixed = NULL,
```

```

    gmc = NULL,
    centering = NULL,
    std = FALSE,
    view = NULL,
    factors = NULL,
    contrasts = "treatment",
    missing = c("listwise", "fiml"),
    WAIC = FALSE,
    ...
  )

```

### Arguments

|           |                                                                                                                                                                                                                                                                                                                |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula   | Formula                                                                                                                                                                                                                                                                                                        |
| data      | Optional data frame. If omitted, variables are resolved from the formula environment. In this mode, use bare variable names; formulas using \$, [, or . require an explicit data argument.                                                                                                                     |
| family    | Character string of the distribution family (e.g., "gaussian", "binomial", "poisson")                                                                                                                                                                                                                          |
| prior     | An object of class "rtmb_prior". Use 'prior_flat()' for no prior, 'prior_normal()' for default normal/exponential priors, 'prior_jzs()' for continuous-family JZS priors, or 'prior_weak()', 'prior_rhs()', 'prior_ssp()' for weakly informative or regularized Bayesian inference. Default is 'prior_flat()'. |
| y_range   | Theoretical minimum and maximum values of the response variable as a vector c(min, max). Specifying this automatically enables weakly informative priors.                                                                                                                                                      |
| init      | List of initial values                                                                                                                                                                                                                                                                                         |
| fixed     | Optional named list of fixed values for specific parameters.                                                                                                                                                                                                                                                   |
| gmc       | Character vector of variable names for GMC                                                                                                                                                                                                                                                                     |
| centering | Alias for 'gmc'.                                                                                                                                                                                                                                                                                               |
| std       | Logical; if 'TRUE', also report post-hoc standardized fixed-effect coefficients. All fixed-effect design-matrix columns are standardized.                                                                                                                                                                      |
| view      | Optional character vector of parameter names to show first in summary.                                                                                                                                                                                                                                         |
| factors   | Character vector of variable names to be treated as factors.                                                                                                                                                                                                                                                   |
| contrasts | Character string specifying the contrast type ("treatment" or "sum").                                                                                                                                                                                                                                          |
| missing   | Missing value handling strategy: "listwise".                                                                                                                                                                                                                                                                   |
| WAIC      | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                                      |
| ...       | Additional arguments passed to rtmb_glmr().                                                                                                                                                                                                                                                                    |

### Examples

```

# --- 1. Linear Regression (rtmb_lm) ---
# Fit a linear regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_lm <- rtmb_lm(sat ~ talk + perf, data = debate)
map_lm <- fit_lm$optimize()

```

```

map_lm$summary()

# --- 2. Generalized Linear Model (rtmb_glm) ---
# Fit a logistic regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glm <- rtmb_glm(cond ~ talk + sat, data = debate, family = "bernoulli")
map_glm <- fit_glm$optimize()
map_glm$summary()

# --- 3. Generalized Linear Mixed Model (rtmb_glmer) ---
# Fit a linear mixed-effects model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glmer <- rtmb_glmer(talk ~ cond + (1 | group), data = debate, family = "gaussian")

# MAP estimation using Laplace approximation for random effects
map_glmer <- fit_glmer$optimize(laplace = TRUE)
map_glmer$summary()

# MCMC sampling (chains and iterations reduced for faster execution)

mcmc_glmer <- fit_glmer$sample(sampling = 500, warmup = 500, chains = 2)
mcmc_glmer$summary()

# --- 4. Linear Mixed Model (rtmb_lmer) ---
# A convenient wrapper for Gaussian mixed models (identical to rtmb_glmer with family="gaussian")
fit_lmer <- rtmb_lmer(sat ~ talk + (1 | group), data = debate)
map_lmer <- fit_lmer$optimize()
map_lmer$summary()

# --- 5. Regularized Regression (Variable Selection) ---
# You can apply regularization to the fixed effects to shrink noise variables towards zero.
# Use prior = prior_rhs() for the Regularized Horseshoe prior,
# or prior_ssp() for the Spike-and-Slab prior.
# Note: When using regularization, you must specify 'y_range' (the theoretical minimum and maximum
# values of the response variable) to automatically set up the required weakly informative priors.

# Fit a linear regression using debate predictors with the Horseshoe prior
fit_rhs <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_rhs(),
  y_range = c(1, 5)
)
map_rhs <- fit_rhs$optimize()
# Summarize only the fixed effects (slopes)
map_rhs$summary("b")

# Fit a linear regression with the Spike-and-Slab prior
fit_ssp <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_ssp(),

```

```

    y_range = c(1, 5)
  )
  map_ssp <- fit_ssp$optimize()
  map_ssp$summary("b")

```

---

rtmb\_glmer

*RTMB-based GLMM wrapper function*


---

## Description

RTMB-based GLMM wrapper function

## Usage

```

rtmb_glmer(
  formula,
  data = NULL,
  family = "gaussian",
  laplace = FALSE,
  prior = prior_flat(),
  y_range = NULL,
  init = NULL,
  fixed = NULL,
  gmc = NULL,
  centering = NULL,
  cwc = NULL,
  std = FALSE,
  view = NULL,
  within = NULL,
  factors = NULL,
  contrasts = "treatment",
  sigma_by = NULL,
  resid_corr = NULL,
  resid_time = NULL,
  resid_group = NULL,
  generate = NULL,
  missing = c("listwise", "fiml"),
  WAIC = FALSE,
  .force_sum = FALSE
)

```

## Arguments

|         |                                                                                                                                                                                             |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | lme4-style formula (e.g., $Y \sim X + (1   \text{GID})$ )                                                                                                                                   |
| data    | Optional data frame. If omitted, variables are resolved from the formula environment. In this mode, use bare variable names; formulas using \$, [[, or . require an explicit data argument. |

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| family      | Character string of the distribution family (e.g., "gaussian", "binomial", "poisson", "ordered", "sequential")                                                                                                                                                                                                                                                                                                                     |
| laplace     | Logical; whether to marginalize random effects using Laplace approximation                                                                                                                                                                                                                                                                                                                                                         |
| prior       | An object of class "rtmb_prior" specifying the prior distribution. Use 'prior_flat()', 'prior_normal()', 'prior_weak()', 'prior_rhs()', or 'prior_ssp()'. 'prior_jzs()' is available for continuous families. Default is 'prior_flat()'.                                                                                                                                                                                           |
| y_range     | Theoretical minimum and maximum values of the response variable as a vector c(min, max). Required when using weakly informative or regularized priors with continuous models.                                                                                                                                                                                                                                                      |
| init        | List of initial values (generated automatically based on glm if omitted)                                                                                                                                                                                                                                                                                                                                                           |
| fixed       | Optional named list of fixed values for specific parameters.                                                                                                                                                                                                                                                                                                                                                                       |
| gmc         | Character vector of variable names for Grand Mean Centering (GMC). If "all", all numeric variables are centered.                                                                                                                                                                                                                                                                                                                   |
| centering   | Alias for 'gmc'.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| cwc         | List for Centering Within Cluster (CWC). Should contain cluster (group variable) and pars (variable names to center). You can also use cwc = list(ID, "x") or cwc = list(ID, "all"); "all" centers all numeric fixed-effect variables within the cluster.                                                                                                                                                                          |
| std         | Logical; if 'TRUE', add post-hoc standardized fixed-effect coefficients as 'b_std'. Every column of the fixed-effect design matrix is standardized, including columns generated from factors and interactions. For Gaussian models, coefficients are scaled by both the predictor and response standard deviations. For other families, only the predictor standard deviations are used, so coefficients remain on the link scale. |
| view        | Character vector of parameter names to prioritize in summary.                                                                                                                                                                                                                                                                                                                                                                      |
| within      | Optional list for wide-to-long conversion.                                                                                                                                                                                                                                                                                                                                                                                         |
| factors     | Character vector of variable names to be treated as factors.                                                                                                                                                                                                                                                                                                                                                                       |
| contrasts   | Character string specifying the contrast type ("treatment" or "sum").                                                                                                                                                                                                                                                                                                                                                              |
| sigma_by    | Character vector specifying variables to group residual variance by (heteroscedasticity).                                                                                                                                                                                                                                                                                                                                          |
| resid_corr  | Residual correlation structure: "ar1" (Autoregressive), "cs" (Compound Symmetry), "toep" (Toeplitz), or "un" (Unstructured).                                                                                                                                                                                                                                                                                                       |
| resid_time  | Variable name for time points in residual correlation.                                                                                                                                                                                                                                                                                                                                                                             |
| resid_group | Variable name for grouping in residual correlation.                                                                                                                                                                                                                                                                                                                                                                                |
| generate    | Optional expression for generated quantities.                                                                                                                                                                                                                                                                                                                                                                                      |
| missing     | Missing value handling strategy: "listwise".                                                                                                                                                                                                                                                                                                                                                                                       |
| WAIC        | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                                                                                                                                                          |
| .force_sum  | Logical; internal use only.                                                                                                                                                                                                                                                                                                                                                                                                        |

## Examples

```
# --- 1. Linear Regression (rtmb_lm) ---
# Fit a linear regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_lm <- rtmb_lm(sat ~ talk + perf, data = debate)
map_lm <- fit_lm$optimize()
map_lm$summary()

# --- 2. Generalized Linear Model (rtmb_glm) ---
# Fit a logistic regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glm <- rtmb_glm(cond ~ talk + sat, data = debate, family = "bernoulli")
map_glm <- fit_glm$optimize()
map_glm$summary()

# --- 3. Generalized Linear Mixed Model (rtmb_glmer) ---
# Fit a linear mixed-effects model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glmer <- rtmb_glmer(talk ~ cond + (1 | group), data = debate, family = "gaussian")

# MAP estimation using Laplace approximation for random effects
map_glmer <- fit_glmer$optimize(laplace = TRUE)
map_glmer$summary()

# MCMC sampling (chains and iterations reduced for faster execution)

mcmc_glmer <- fit_glmer$sample(sampling = 500, warmup = 500, chains = 2)
mcmc_glmer$summary()

# --- 4. Linear Mixed Model (rtmb_lmer) ---
# A convenient wrapper for Gaussian mixed models (identical to rtmb_glmer with family="gaussian")
fit_lmer <- rtmb_lmer(sat ~ talk + (1 | group), data = debate)
map_lmer <- fit_lmer$optimize()
map_lmer$summary()

# --- 5. Regularized Regression (Variable Selection) ---
# You can apply regularization to the fixed effects to shrink noise variables towards zero.
# Use prior = prior_rhs() for the Regularized Horseshoe prior,
# or prior_ssp() for the Spike-and-Slab prior.
# Note: When using regularization, you must specify 'y_range' (the theoretical minimum and maximum
# values of the response variable) to automatically set up the required weakly informative priors.

# Fit a linear regression using debate predictors with the Horseshoe prior
fit_rhs <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_rhs(),
  y_range = c(1, 5)
)
map_rhs <- fit_rhs$optimize()
# Summarize only the fixed effects (slopes)
```

```

map_rhs$summary("b")

# Fit a linear regression with the Spike-and-Slab prior
fit_ssp <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_ssp(),
  y_range = c(1, 5)
)
map_ssp <- fit_ssp$optimize()
map_ssp$summary("b")

```

---

rtmb\_irt

*RTMB-based IRT (Item Response Theory) Wrapper*


---

## Description

Performs Item Response Theory modeling (1PL, 2PL, 3PL, and Graded Response Model) using RTMB. Missing values (NA) in the data are automatically removed internally for efficient computation.

## Usage

```

rtmb_irt(
  data,
  model = c("2PL", "1PL", "3PL"),
  type = c("binary", "ordered"),
  prior = prior_flat(),
  init = NULL,
  fixed = NULL,
  view = NULL,
  missing = c("listwise", "fiml"),
  WAIC = FALSE
)

```

## Arguments

|                    |                                                                                                                                                                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>  | A data frame or matrix of item responses (N persons x J items).                                                                                                                                                                                                                                                             |
| <code>model</code> | Character string for the model type: "1PL", "2PL", or "3PL".                                                                                                                                                                                                                                                                |
| <code>type</code>  | Character string for the data type: "binary" or "ordered".                                                                                                                                                                                                                                                                  |
| <code>prior</code> | Prior configuration: 'prior_flat()', 'prior_normal()', or 'prior_weak()'. Hyperparameters can be specified within these functions (e.g., 'prior_normal(b_sd = 5)'). Available parameters for IRT: 'a_rate' (discrimination; defaults to 1/2 with 'prior_normal()'), 'b_sd' (difficulty), and 'c_alpha'/'c_beta' (guessing). |
| <code>init</code>  | List of initial values.                                                                                                                                                                                                                                                                                                     |
| <code>fixed</code> | A named list of parameter values to fix (optional).                                                                                                                                                                                                                                                                         |

|         |                                                                                                        |
|---------|--------------------------------------------------------------------------------------------------------|
| view    | Character vector of parameter names to prioritize in summary.                                          |
| missing | Missing value handling strategy: "listwise" (default) or "fiml" (Full Information Maximum Likelihood). |
| WAIC    | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                              |

## Examples

```
# --- 1. Binary Data (e.g., correct/incorrect answers) ---
# Simulate binary response data for 100 persons and 5 items
set.seed(123)
bin_data <- matrix(rbinom(500, size = 1, prob = 0.6), nrow = 100, ncol = 5)
colnames(bin_data) <- paste0("Item", 1:5)

# Introduce some missing values (NA) to demonstrate automatic handling
#bin_data[sample(1:500, 10)] <- NA

# Fit a 2-Parameter Logistic (2PL) model
fit_2pl <- rtmb_irt(data = bin_data, model = "2PL", type = "binary")

# Maximum A Posteriori (MAP) estimation
map_2pl <- fit_2pl$optimize()
map_2pl$summary()

# --- 2. Ordered Data (e.g., Likert scale) ---
# Simulate ordered response data (categories 1 to 5) with a true latent structure
set.seed(123)
N <- 100
J <- 5
K <- 4

# True parameters
theta <- rnorm(N, 0, 1)          # Person abilities (latent traits)
a <- runif(J, 0.8, 2.0)         # Item discriminations

# Item-specific thresholds (must be strictly increasing)
b <- matrix(NA, nrow = J, ncol = K - 1)
for (j in 1:J) {
  b[j, ] <- sort(c(-1.5, 0, 1.5) + rnorm(3, 0, 0.2))
}

ord_data <- matrix(NA, nrow = N, ncol = J)
colnames(ord_data) <- paste0("Item", 1:J)

# Generate responses based on the Graded Response Model
for (i in 1:N) {
  for (j in 1:J) {
    # Latent continuous response (eta + standard logistic noise)
    eta <- a[j] * theta[i]
    y_star <- eta + rlogis(1)

    # Apply thresholds to determine the observed category (1 to 4)
    category <- 1
  }
}
```

```

    for (k in 1:(K - 1)) {
      if (y_star > b[j, k]) category <- k + 1
    }
    ord_data[i, j] <- category
  }
}

# Fit a Graded Response Model (2PL for ordered data)
fit_ord <- rtmb_irt(data = ord_data, model = "2PL", type = "ordered")
fit_ord$print_code()

map_ord <- fit_ord$optimize()
map_ord$summary()

```

---

rtmb\_lm

*RTMB-based Linear Regression wrapper function*


---

## Description

RTMB-based Linear Regression wrapper function

## Usage

```

rtmb_lm(
  formula,
  data = NULL,
  prior = prior_flat(),
  y_range = NULL,
  init = NULL,
  fixed = NULL,
  gmc = NULL,
  centering = NULL,
  std = FALSE,
  view = NULL,
  factors = NULL,
  contrasts = "treatment",
  missing = c("listwise", "fiml"),
  WAIC = FALSE,
  ...
)

```

## Arguments

|         |                                                                                                                                                                                             |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | Formula (e.g., $Y \sim X1 + X2$ )                                                                                                                                                           |
| data    | Optional data frame. If omitted, variables are resolved from the formula environment. In this mode, use bare variable names; formulas using \$, [], or . require an explicit data argument. |

|           |                                                                                                                                                                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| prior     | An object of class "rtmb_prior". Use 'prior_flat()' for no prior, 'prior_normal()' for default normal/exponential priors, 'prior_jzs()' for JZS regression priors, or 'prior_weak()', 'prior_rhs()', 'prior_ssp()' for weakly informative or regularized Bayesian inference. Default is 'prior_flat()'. |
| y_range   | Theoretical minimum and maximum values of the response variable as a vector c(min, max). Specifying this automatically enables weakly informative priors.                                                                                                                                               |
| init      | List of initial values.                                                                                                                                                                                                                                                                                 |
| fixed     | Optional named list of fixed values for specific parameters.                                                                                                                                                                                                                                            |
| gmc       | Character vector of variable names for GMC                                                                                                                                                                                                                                                              |
| centering | Alias for 'gmc'.                                                                                                                                                                                                                                                                                        |
| std       | Logical; if 'TRUE', also report post-hoc standardized fixed-effect coefficients. All fixed-effect design-matrix columns are standardized.                                                                                                                                                               |
| view      | Optional character vector of parameter names to show first in summary.                                                                                                                                                                                                                                  |
| factors   | Character vector of variable names to be treated as factors.                                                                                                                                                                                                                                            |
| contrasts | Character string specifying the contrast type ("treatment" or "sum").                                                                                                                                                                                                                                   |
| missing   | Missing value handling strategy: "listwise".                                                                                                                                                                                                                                                            |
| WAIC      | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                               |
| ...       | Additional arguments passed to rtmb_glmr().                                                                                                                                                                                                                                                             |

## Examples

```
# --- 1. Linear Regression (rtmb_lm) ---
# Fit a linear regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_lm <- rtmb_lm(sat ~ talk + perf, data = debate)
map_lm <- fit_lm$optimize()
map_lm$summary()

# --- 2. Generalized Linear Model (rtmb_glm) ---
# Fit a logistic regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glm <- rtmb_glm(cond ~ talk + sat, data = debate, family = "bernoulli")
map_glm <- fit_glm$optimize()
map_glm$summary()

# --- 3. Generalized Linear Mixed Model (rtmb_glmr) ---
# Fit a linear mixed-effects model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glmr <- rtmb_glmr(talk ~ cond + (1 | group), data = debate, family = "gaussian")

# MAP estimation using Laplace approximation for random effects
map_glmr <- fit_glmr$optimize(laplace = TRUE)
map_glmr$summary()

# MCMC sampling (chains and iterations reduced for faster execution)

mcmc_glmr <- fit_glmr$sample(sampling = 500, warmup = 500, chains = 2)
```

```

mcmc_glmmer$summary()

# --- 4. Linear Mixed Model (rtmb_lmer) ---
# A convenient wrapper for Gaussian mixed models (identical to rtmb_glmmer with family="gaussian")
fit_lmer <- rtmb_lmer(sat ~ talk + (1 | group), data = debate)
map_lmer <- fit_lmer$optimize()
map_lmer$summary()

# --- 5. Regularized Regression (Variable Selection) ---
# You can apply regularization to the fixed effects to shrink noise variables towards zero.
# Use prior = prior_rhs() for the Regularized Horseshoe prior,
# or prior_ssp() for the Spike-and-Slab prior.
# Note: When using regularization, you must specify 'y_range' (the theoretical minimum and maximum
# values of the response variable) to automatically set up the required weakly informative priors.

# Fit a linear regression using debate predictors with the Horseshoe prior
fit_rhs <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_rhs(),
  y_range = c(1, 5)
)
map_rhs <- fit_rhs$optimize()
# Summarize only the fixed effects (slopes)
map_rhs$summary("b")

# Fit a linear regression with the Spike-and-Slab prior
fit_ssp <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_ssp(),
  y_range = c(1, 5)
)
map_ssp <- fit_ssp$optimize()
map_ssp$summary("b")

```

---

rtmb\_lmer

*RTMB-based Linear Mixed Model (LMM) wrapper function*


---

## Description

RTMB-based Linear Mixed Model (LMM) wrapper function

## Usage

```

rtmb_lmer(
  formula,
  data = NULL,
  laplace = TRUE,

```

```

prior = prior_flat(),
y_range = NULL,
init = NULL,
fixed = NULL,
gmc = NULL,
centering = NULL,
cwc = NULL,
std = FALSE,
view = NULL,
sigma_by = NULL,
factors = NULL,
contrasts = "treatment",
resid_corr = NULL,
resid_time = NULL,
resid_group = NULL,
within = NULL,
missing = c("listwise", "fiml"),
WAIC = FALSE,
...
)

```

### Arguments

|           |                                                                                                                                                                                                                                                                                                         |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula   | Formula                                                                                                                                                                                                                                                                                                 |
| data      | Optional data frame. If omitted, variables are resolved from the formula environment. In this mode, use bare variable names; formulas using \$, [[, or . require an explicit data argument.                                                                                                             |
| laplace   | Logical; whether to marginalize random effects using Laplace approximation                                                                                                                                                                                                                              |
| prior     | An object of class "rtmb_prior". Use 'prior_flat()' for no prior, 'prior_normal()' for default normal/exponential priors, 'prior_jzs()' for JZS regression priors, or 'prior_weak()', 'prior_rhs()', 'prior_ssp()' for weakly informative or regularized Bayesian inference. Default is 'prior_flat()'. |
| y_range   | Theoretical minimum and maximum values of the response variable                                                                                                                                                                                                                                         |
| init      | Initial values                                                                                                                                                                                                                                                                                          |
| fixed     | Optional named list of fixed values for specific parameters.                                                                                                                                                                                                                                            |
| gmc       | Character vector of variable names for GMC                                                                                                                                                                                                                                                              |
| centering | Alias for 'gmc'.                                                                                                                                                                                                                                                                                        |
| cwc       | List for CWC                                                                                                                                                                                                                                                                                            |
| std       | Logical; if 'TRUE', also report post-hoc standardized fixed-effect coefficients. All fixed-effect design-matrix columns are standardized.                                                                                                                                                               |
| view      | Character vector of parameter names to prioritize in summary.                                                                                                                                                                                                                                           |
| sigma_by  | Character vector specifying variables to group residual variance by (heteroscedasticity).                                                                                                                                                                                                               |
| factors   | Character vector of variable names to be treated as factors.                                                                                                                                                                                                                                            |
| contrasts | Character string specifying the contrast type ("treatment" or "sum").                                                                                                                                                                                                                                   |

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>resid_corr</code>  | Residual correlation structure (e.g., "ar1", "cs", "un", "toep").                                                                                                                                                                                                                                                                                                                                                                       |
| <code>resid_time</code>  | Variable name for time points in residual correlation.                                                                                                                                                                                                                                                                                                                                                                                  |
| <code>resid_group</code> | Variable name for grouping in residual correlation.                                                                                                                                                                                                                                                                                                                                                                                     |
| <code>within</code>      | Optional list for wide-to-long conversion. For repeated measures data in wide format, specify the factor names and their levels, e.g., <code>list(Time = 4)</code> or <code>list(A = 2, B = 3)</code> . The total number of levels must match the number of columns in <code>cbind()</code> on the LHS. If omitted and the LHS is <code>cbind()</code> , the within-factor name is inferred from RHS variables not present in the data. |
| <code>missing</code>     | Missing value handling strategy: "listwise".                                                                                                                                                                                                                                                                                                                                                                                            |
| <code>WAIC</code>        | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                                                                                                                                                               |
| <code>...</code>         | Additional arguments passed to <code>rtmb_glmmer()</code> .                                                                                                                                                                                                                                                                                                                                                                             |

## Value

RTMB\_Model object

## Examples

```
# --- 1. Linear Regression (rtmb_lm) ---
# Fit a linear regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_lm <- rtmb_lm(sat ~ talk + perf, data = debate)
map_lm <- fit_lm$optimize()
map_lm$summary()

# --- 2. Generalized Linear Model (rtmb_glm) ---
# Fit a logistic regression model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glm <- rtmb_glm(cond ~ talk + sat, data = debate, family = "bernoulli")
map_glm <- fit_glm$optimize()
map_glm$summary()

# --- 3. Generalized Linear Mixed Model (rtmb_glmmer) ---
# Fit a linear mixed-effects model using the debate dataset
data(debate, package = "BayesRTMB")
fit_glmmer <- rtmb_glmmer(talk ~ cond + (1 | group), data = debate, family = "gaussian")

# MAP estimation using Laplace approximation for random effects
map_glmmer <- fit_glmmer$optimize(laplace = TRUE)
map_glmmer$summary()

# MCMC sampling (chains and iterations reduced for faster execution)

mcmc_glmmer <- fit_glmmer$sample(sampling = 500, warmup = 500, chains = 2)
mcmc_glmmer$summary()

# --- 4. Linear Mixed Model (rtmb_lmer) ---
# A convenient wrapper for Gaussian mixed models (identical to rtmb_glmmer with family="gaussian")
fit_lmer <- rtmb_lmer(sat ~ talk + (1 | group), data = debate)
```

```

map_lmer <- fit_lmer$optimize()
map_lmer$summary()

# --- 5. Regularized Regression (Variable Selection) ---
# You can apply regularization to the fixed effects to shrink noise variables towards zero.
# Use prior = prior_rhs() for the Regularized Horseshoe prior,
# or prior_ssp() for the Spike-and-Slab prior.
# Note: When using regularization, you must specify 'y_range' (the theoretical minimum and maximum
# values of the response variable) to automatically set up the required weakly informative priors.

# Fit a linear regression using debate predictors with the Horseshoe prior
fit_rhs <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_rhs(),
  y_range = c(1, 5)
)
map_rhs <- fit_rhs$optimize()
# Summarize only the fixed effects (slopes)
map_rhs$summary("b")

# Fit a linear regression with the Spike-and-Slab prior
fit_ssp <- rtmb_lm(
  sat ~ talk + perf + skill,
  data = debate,
  prior = prior_ssp(),
  y_range = c(1, 5)
)
map_ssp <- fit_ssp$optimize()
map_ssp$summary("b")

```

---

rtmb\_loglinear

*RTMB-based Log-linear analysis (Poisson regression)*


---

## Description

Performs Bayesian or Frequentist log-linear analysis (Poisson regression) on a contingency table or raw data.

## Usage

```

rtmb_loglinear(
  formula,
  data,
  prior = prior_flat(),
  y_range = NULL,
  fixed = NULL,
  WAIC = FALSE,
  ...
)

```

**Arguments**

|         |                                                                                                                                         |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------|
| formula | A formula (e.g., '~ A + B + A:B') or a contingency table.                                                                               |
| data    | A data frame (required if 'formula' is used).                                                                                           |
| prior   | An object of class "rtmb_prior" specifying the prior distribution.                                                                      |
| y_range | Optional theoretical minimum and maximum values of the count response. If supplied with the default flat prior, 'prior_weak()' is used. |
| fixed   | Optional named list of fixed values for specific parameters.                                                                            |
| WAIC    | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                               |
| ...     | Additional arguments passed to 'rtmb_glm()'.                                                                                            |

**Value**

An 'RTMB\_Model' object.

**Examples**

```
# Create a contingency table
tab <- matrix(c(10, 20, 30, 40), nrow = 2)
dimnames(tab) <- list(A = c("A1", "A2"), B = c("B1", "B2"))

# Fit a log-linear model (independence model: ~ A + B)
fit_log <- rtmb_loglinear(~ A + B, data = tab)

# MAP estimation
map_log <- fit_log$optimize()
map_log$summary()
```

---

rtmb\_lrt

---

*Fit a Latent Rank Theory (LRT) Model*


---

**Description**

Fits a Latent Rank Theory model, which is a mixture model with ordered ranks and Gaussian Process smoothing on the mean profiles.

**Usage**

```
rtmb_lrt(
  formula,
  k = 3,
  data = NULL,
  rank_coords = NULL,
  covariance = c("diagonal", "diagonal_equal", "full", "full_equal", "full_equal_corr"),
  magnitude = NULL,
  smoothing = NULL,
  noise = 0.01,
```

```

    prob_smoothing = FALSE,
    link = c("ordered", "sequential"),
    prior = prior_flat(),
    y_range = NULL,
    fixed = NULL,
    two_stage = FALSE,
    WAIC = FALSE,
    ...
  )

```

## Arguments

|                |                                                                                                                                                                                                                                                                                                             |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula        | A formula specifying the response variable(s).                                                                                                                                                                                                                                                              |
| k              | Number of ranks (mixture components).                                                                                                                                                                                                                                                                       |
| data           | A data frame containing the variables.                                                                                                                                                                                                                                                                      |
| rank_coords    | Optional numeric vector of coordinates for each rank. Default is 1:k.                                                                                                                                                                                                                                       |
| covariance     | Covariance structure: "diagonal", "diagonal_equal", "full", "full_equal", or "full_equal_corr".                                                                                                                                                                                                             |
| magnitude      | Signal standard deviation for the GP prior. If NULL, it is estimated.                                                                                                                                                                                                                                       |
| smoothing      | Length-scale for the GP prior. If NULL, it is estimated.                                                                                                                                                                                                                                                    |
| noise          | Measurement noise for the GP prior (default is 0.01).                                                                                                                                                                                                                                                       |
| prob_smoothing | Logical; whether to apply smoothing to the class membership probabilities.                                                                                                                                                                                                                                  |
| link           | Link function for class probabilities: "ordered" or "sequential".                                                                                                                                                                                                                                           |
| prior          | Prior configuration: 'prior_flat()', 'prior_normal()', 'prior_weak()', 'prior_rhs()', or 'prior_ssp()'. Default is 'prior_flat()'. If 'y_range' is supplied with the default flat prior, the wrapper automatically switches to 'prior_weak()'.                                                              |
| y_range        | Optional theoretical range of the response variables. Use a numeric vector 'c(min, max)' for a common range, or a P x 2 matrix/list for response-specific ranges. Specifying this automatically enables 'prior_weak()' when 'prior' is 'prior_flat()' and calibrates the residual standard-deviation prior. |
| fixed          | Optional named list of fixed values for specific parameters.                                                                                                                                                                                                                                                |
| two_stage      | Logical; if TRUE, estimate the latent-rank measurement model first and then estimate the rank regression with delta-method uncertainty propagation. Currently supported for '\$optimize()' only.                                                                                                            |
| WAIC           | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                                   |
| ...            | Additional arguments passed to 'rtmb_model'.                                                                                                                                                                                                                                                                |

## Value

A RTMB\_Model object.

rtmb\_mdu

*RTMB-based Multidimensional Unfolding Wrapper***Description**

Fits a multidimensional unfolding model for preference/rating data. Rows are persons or observations and columns are stimuli/items. The model represents both row scores ('theta') and item locations ('delta') in a shared D-dimensional space.

**Usage**

```
rtmb_mdu(
  data,
  ndim = 2,
  distance = c("euclidean", "squared"),
  alpha = c("random", "fix"),
  lambda = c("fix", "random"),
  method = c("rating", "Best", "Best-Worst", "MDS"),
  sets = NULL,
  prior = prior_flat(),
  y_range = NULL,
  init = NULL,
  fixed = NULL,
  view = NULL,
  distance_eps = 1e-04,
  missing = c("listwise", "fiml"),
  WAIC = FALSE
)
```

**Arguments**

|          |                                                                                                                                                                                                                                                                        |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data     | Numeric matrix or data frame (N rows x M items) for 'method = "rating"'. For choice methods, a list containing 'Best' and, for 'method = "Best-Worst"', 'Worst'. For 'method = "Best-Worst"', a single matrix/data frame is treated as pre-coded 'Y_dif' pair indices. |
| ndim     | Number of unfolding dimensions.                                                                                                                                                                                                                                        |
| distance | Character; "euclidean" uses Euclidean distance (default), while "squared" uses squared Euclidean distance.                                                                                                                                                             |
| alpha    | Character; "random" estimates item-specific alpha values as random effects (default), while "fix" estimates a single common alpha.                                                                                                                                     |
| lambda   | Character; for choice methods, "fix" estimates a common inverse-temperature parameter (default), while "random" estimates person-specific inverse-temperature parameters with a log-normal hierarchy.                                                                  |
| method   | Character; "rating" for continuous ratings, "Best" for best-only choice tasks, "Best-Worst" for best-worst choice tasks, or "MDS" for fitting a multidimensional scaling model to a distance matrix.                                                                   |

|              |                                                                                                                                                                                                                                                                                                      |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sets         | Matrix or data frame of presented item sets (P tasks x C items) for choice methods.                                                                                                                                                                                                                  |
| prior        | Prior configuration: 'prior_flat()', 'prior_normal()', or 'prior_weak()'. 'prior_flat()' creates a maximum-likelihood model suitable for 'classic()'. The latent coordinates 'delta' and 'theta' are always treated as random effects with normal scale priors, similarly to IRT ability parameters. |
| y_range      | Optional response range for 'method = "rating"'. If supplied with the default flat prior, 'prior_weak()' is used. For choice methods ("Best" and "Best-Worst"), 'prior_weak()' behaves like 'prior_normal()' because there is no observed rating scale.                                              |
| init         | Optional named list of initial values.                                                                                                                                                                                                                                                               |
| fixed        | Optional named list of parameter values to fix.                                                                                                                                                                                                                                                      |
| view         | Character vector of parameter names to prioritize in summaries.                                                                                                                                                                                                                                      |
| distance_eps | Small positive constant added to the distance.                                                                                                                                                                                                                                                       |
| missing      | Missing value handling strategy: "listwise" (default) or "fiml" (Full Information Maximum Likelihood).                                                                                                                                                                                               |
| WAIC         | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                            |

### Value

An 'RTMB\_Model' object.

### Examples

```
# Simulate rating data for Multidimensional Unfolding (MDU)
set.seed(123)
N <- 50 # Number of persons
M <- 10 # Number of items
D <- 2  # Number of dimensions

# True person and item coordinates in a 2D space
theta <- matrix(rnorm(N * D), N, D)
delta <- matrix(rnorm(M * D), M, D)

# Generate distance-like ratings (smaller means more preferred)
Y <- matrix(NA, N, M)
for(i in 1:N) {
  for(j in 1:M) {
    Y[i, j] <- sum((theta[i,] - delta[j,])^2) + rnorm(1, 0, 0.5)
  }
}

# Fit a 2-dimensional MDU model
fit_mdu <- rtmb_mdu(Y, ndim = 2, distance = "squared", method = "rating")

# MAP estimation
map_mdu <- fit_mdu$optimize()
map_mdu$summary()
```

```
# Note: MDU models have many parameters, so MCMC sampling might take time.

# mcmc_mdu <- fit_mdu$sample(sampling = 500, warmup = 500, chains = 2)
```

---

|                |                                              |
|----------------|----------------------------------------------|
| rtmb_mediation | <i>RTMB-based Mediation Analysis Wrapper</i> |
|----------------|----------------------------------------------|

---

## Description

‘rtmb\_mediation’ performs mediation analysis by simultaneously estimating multiple GLM regression equations. It automatically identifies mediation paths and calculates indirect, direct, and total effects.

## Usage

```
rtmb_mediation(
  formula,
  data,
  family = "gaussian",
  prior = prior_flat(),
  y_range = NULL,
  fixed = NULL,
  view = NULL,
  WAIC = FALSE,
  gmc = NULL,
  centering = NULL,
  cwc = NULL,
  ...
)
```

## Arguments

|         |                                                                                                                                                                                      |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula | A list of formulas defining the regression paths (e.g., ‘list(M ~ X, Y ~ X + M)’). Each equation may optionally include one random intercept, such as ‘(1   ID)’.                    |
| data    | A data frame containing the variables.                                                                                                                                               |
| family  | A single character string or a list of character strings specifying the error distribution for each equation (e.g., ‘family = list("gaussian", "binomial")’). Default is "gaussian". |
| prior   | An object of class "rtmb_prior" specifying the prior distribution.                                                                                                                   |
| y_range | Theoretical minimum and maximum values of the response variable.                                                                                                                     |
| fixed   | A named list of parameter values to fix (optional).                                                                                                                                  |
| view    | Character vector of parameter names to prioritize in summary.                                                                                                                        |
| WAIC    | Logical; if TRUE, add pointwise ‘log_lik’ to the generate block for WAIC.                                                                                                            |

|           |                                                                                                                                                                   |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| gmc       | Character vector naming predictors to grand-mean center, or ""all"" to center all numeric predictors used by the mediation equations.                             |
| centering | Alias for 'gmc'.                                                                                                                                                  |
| cwc       | Centering-within-cluster specification. Use, for example, 'list(cluster = ID, pars = c("X", "M"))' or 'list(ID, "X")'. Cluster means are not added automatically. |
| ...       | Reserved; unused arguments are rejected.                                                                                                                          |

## Details

The function identifies mediation paths by looking for variables that are responses in one equation and predictors in another. Indirect effects are calculated as the product of coefficients along these paths ( $a * b$ ).

Random intercepts may be included in all equations or in only a subset of equations. When more than one equation contains a random intercept, all random intercepts must use the same grouping variable and their correlations are estimated jointly. Random slopes, multiple random-effect terms within an equation, and different grouping variables across equations are not yet supported. The mediation-specific classical bootstrap is currently unavailable for random-intercept and CWC models.

'gmc' (or its alias 'centering') and 'cwc' are applied to predictor uses of the selected variables before each equation's model matrix is constructed. Their response uses remain on the original scale. Thus, if 'M' is the response in one equation and a predictor in another, 'cwc = list(ID, "M")' centers 'M' only in the latter role. When both GMC and CWC target the same variable, GMC is applied first. Between-cluster means must be created by the user and included explicitly in the formulas.

**Uncertainty Estimation:** When using '\$optimize(ci\_method = "sampling")', the function provides asymmetric confidence intervals for indirect effects based on the distribution of products, which is more accurate than the standard Sobel test (Delta Method).

## Value

An 'RTMB\_Model' object.

## Examples

```
# Simulate mediation data
set.seed(123)
N <- 100
X <- rnorm(N)
# M is influenced by X
M <- 0.5 * X + rnorm(N, 0, 0.5)
# Y is influenced by both X and M
Y <- 0.3 * X + 0.8 * M + rnorm(N, 0, 0.5)
dat <- data.frame(X, M, Y)

# Fit a mediation model
# The formula list specifies the two regression equations
fit_med <- rtmb_mediation(list(M ~ X, Y ~ X + M), data = dat)
```

```
# Maximum A Posteriori (MAP) estimation
map_med <- fit_med$optimize()
# The summary automatically calculates indirect, direct, and total effects
map_med$summary()
```

---

rtmb\_mixture

*Mixture Model Wrapper for RTMB*


---

## Description

Provides a user-friendly interface for fitting Gaussian mixture models with optional covariates on class membership probabilities and various covariance structures.

## Usage

```
rtmb_mixture(
  formula,
  k = 2,
  data = NULL,
  covariance = c("diagonal", "diagonal_equal", "full", "full_equal", "full_equal_corr"),
  prior = prior_flat(),
  y_range = NULL,
  fixed = NULL,
  WAIC = FALSE,
  ...
)
```

## Arguments

|            |                                                                                                                                                                                                                                                                                                                                 |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| formula    | A formula specifying the response variable(s). For multivariate, use <code>cbind(y1, y2) ~ 1</code> .                                                                                                                                                                                                                           |
| k          | Number of mixture components.                                                                                                                                                                                                                                                                                                   |
| data       | A data frame containing the variables in the model.                                                                                                                                                                                                                                                                             |
| covariance | Covariance structure: "diagonal" (default), "diagonal_equal", "full", "full_equal", or "full_equal_corr".                                                                                                                                                                                                                       |
| prior      | Prior configuration: 'prior_flat()', 'prior_normal()', 'prior_weak()', 'prior_rhs()', or 'prior_ssp()'. Default is 'prior_flat()'. If 'y_range' is supplied with the default flat prior, the wrapper automatically switches to 'prior_weak()'.                                                                                  |
| y_range    | Optional theoretical range of the response variables. Use a numeric vector 'c(min, max)' for a common range, or a P x 2 matrix/list for response-specific ranges. Specifying this automatically enables 'prior_weak()' when 'prior' is 'prior_flat()' and calibrates the component mean and residual standard-deviation priors. |
| fixed      | Optional named list of fixed values for specific parameters.                                                                                                                                                                                                                                                                    |
| WAIC       | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC.                                                                                                                                                                                                                                                       |
| ...        | Reserved; unused arguments are rejected.                                                                                                                                                                                                                                                                                        |

**Value**

A RTMB\_Model object.

**Examples**

```
# Simulate 1D mixture data (2 components)
set.seed(123)
N <- 100
group <- rbinom(N, 1, 0.4)
y <- ifelse(group == 1, rnorm(N, 5, 1), rnorm(N, 0, 1))
dat <- data.frame(y)

# Fit a 1D Gaussian mixture model with 2 components
fit_mix <- rtmb_mixture(y ~ 1, k = 2, data = dat)

# MAP estimation
map_mix <- fit_mix$optimize()
map_mix$summary()
```

---

rtmb\_model

---

*Create an RTMB\_Model Object*


---

**Description**

The ‘rtmb\_model’ function acts as the core constructor for compiling and combining user-defined data with the model code (defined via ‘rtmb\_code’). It generates an ‘RTMB\_Model’ (R6 class) instance, which serves as the foundation for performing Bayesian inference, including Maximum A Posteriori (MAP) estimation, Variational Inference (ADVI), and Markov Chain Monte Carlo (MCMC) sampling.

**Usage**

```
rtmb_model(
  data,
  code,
  par_names = list(),
  init = NULL,
  view = NULL,
  fixed = NULL,
  silent = FALSE,
  gr_test = FALSE
)
```

**Arguments**

|      |                                                                                                                                                                                             |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data | A named list, data frame, matrix, or other object containing the observations used in the model. The original object is available as <code>.data</code> while the setup block is evaluated. |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|                        |                                                                                                                                                                                                                    |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>code</code>      | A model definition block generated by <code>rtmb_code(...)</code> (must include parameters and model blocks).                                                                                                      |
| <code>par_names</code> | A named list of character vectors corresponding to the dimensions of specific parameters (optional).                                                                                                               |
| <code>init</code>      | A list or numeric vector of initial values for parameters (optional). If not specified, initialized randomly.                                                                                                      |
| <code>view</code>      | Character vector of parameter names to be displayed preferentially at the top when outputting results like <code>summary()</code> (optional).                                                                      |
| <code>fixed</code>     | A named list of parameter values to fix (optional). Useful for scoring or plug-in estimation where some parameters (e.g., item parameters) are fixed to known values.                                              |
| <code>silent</code>    | Logical; if TRUE, suppresses diagnostic messages during model creation. Default is FALSE.                                                                                                                          |
| <code>gr_test</code>   | Logical; if TRUE, evaluate the gradient once after the AD tape is built. By default, model creation checks that <code>MakeADFun()</code> can build the AD tape but skips this extra gradient evaluation for speed. |

## Details

**Model Compilation and Pre-checking:** When this function is called, it evaluates the provided data and model blocks. It performs a "sandbox execution" (pre-check) using dummy initial values to dynamically detect common structural errors, such as undefined variables, out-of-bounds indices, or incompatible matrix operations, before proceeding to the computationally expensive Automatic Differentiation (MakeADFun) phase. Cryptic backend errors are caught and translated into user-friendly hints.

Inside the setup block, the read-only reserved name `.data` refers to the original object supplied to `data`. This allows setup code such as `Y <- as.matrix(.data)` for a matrix or data frame, and `S <- as.matrix(.data$sets)` for a named list. The temporary `.data` binding is removed after setup has finished.

**Writing AD-Compatible Code (Important):** To ensure the model is differentiable, you must follow specific syntax rules when writing code within `rtmb_code`. Avoid discrete branching (`if`, `ifelse`) based on parameters and use numerically stable functions. See [rtmb\\_syntax](#) for a detailed guide on Automatic Differentiation requirements.

**Initial Values (`init`):** Initial parameter values can be specified as a flat numeric vector or a named list. If a partial list is provided, the unspecified parameters are automatically initialized with random values drawn from a uniform distribution on the unconstrained scale. If `init = NULL`, all parameters are initialized randomly.

**Parameter Labeling (`par_names`):** By default, vector or matrix parameters are displayed with numeric indices (e.g., `mu[1]`). You can pass a named list of character vectors to `par_names` to assign meaningful labels to specific dimensions (e.g., `mu[Control]`), vastly improving the readability of summary outputs and trace plots.

**Available Inference Methods on the Returned Object:** The returned `RTMB_Model` instance provides the following core methods:

- `$optimize(...)`: Performs Maximum A Posteriori (MAP) or Maximum Likelihood estimation. Returns a `MAP_Fit` object.

- `$sample(...)`: Draws posterior samples using the NUTS (No-U-Turn Sampler) algorithm. Returns an `MCMC_Fit` object.
- `$variational(...)`: Performs Mean-field or Full-rank Automatic Differentiation Variational Inference (ADVI). Returns a `VB_Fit` object.

## Value

An `RTMB_Model` class instance with a compiled and pre-tested automatic differentiation function.

An `RTMB_Model` class instance with a compiled and pre-tested automatic differentiation function.

## Examples

```
# Simulate data for 3 groups
set.seed(123)
N <- 60
group_idx <- sample(1:3, N, replace = TRUE)
group_names <- c("Control", "Treatment_A", "Treatment_B")

# True group means: Control = 0, Treatment_A = 2, Treatment_B = -1
true_means <- c(0, 2, -1)
y <- true_means[group_idx] + rnorm(N, mean = 0, sd = 0.5)

data_list <- list(N = N, K = 3, group_idx = group_idx, y = y)

# Define the model using rtmb_code
model_code <- rtmb_code(
  parameters = {
    mu = Dim(K)          # Vector of length K (group means)
    sigma = Dim(1, lower = 0) # Scalar (residual standard deviation)
  },
  model = {
    # Priors
    for (k in 1:K) mu[k] ~ normal(0, 10)
    sigma ~ exponential(1)

    # Likelihood
    for (i in 1:N) {
      y[i] ~ normal(mu[group_idx[i]], sigma)
    }
  }
)

# --- 1. Basic Model Creation ---
# Create the RTMB_Model object
mod_basic <- rtmb_model(
  data = data_list,
  code = model_code
)

# Perform Maximum A Posteriori (MAP) estimation
map_basic <- mod_basic$optimize()
# The summary displays default parameter names: mu[1], mu[2], mu[3]
```

```

map_basic$summary()

# --- 2. Optional: Adding Custom Parameter Names and initial values ---
# You can optionally use 'par_names' to assign meaningful labels
# to vector or matrix elements for easier interpretation.
mod_named <- rtmb_model(
  data = data_list,
  code = model_code,
  init = list(mu = rep(0, 3), sigma = 1),
  par_names = list(mu = group_names)
)

map_named <- mod_named$optimize()
# The summary now displays: mu[Control], mu[Treatment_A], mu[Treatment_B]
map_named$summary()

```

---

RTMB\_Model-class

*RTMB model object*


---

## Description

An R6 class representing a Bayesian model built with RTMB. This class stores model components and provides methods for building the automatic differentiation object, optimizing the posterior, and drawing posterior samples.

## Public fields

`data` Named list containing observation data.

`par_list` Evaluated list of parameter definitions.

`log_prob` Executable function for likelihood and prior calculations.

`transform` Function for calculating transformed parameters.

`generate` Function for calculating generated quantities.

`par_names` List of variable names corresponding to parameter dimensions.

`pl_full` Complete parameter list with expanded constraints and dimension information used internally.

`formula` Formula used for model generation (if created via a wrapper function).

`requested_contrasts` Character; user-requested contrast setting (e.g. "treatment").

`contrasts` Character; internal contrast setting used for fitting (e.g. "sum").

`raw_data` Data frame; the original data used for the model.

`family` Character string of the distribution family (if created via a wrapper function).

`init` List or numeric vector of initial values.

`view` Character vector of variable names to prioritize in summary.

`code` Original model AST generated by `'rtmb_code()'`.

- `map` A list specifying parameter mappings for fixing values (e.g., via `fixed_model()`).
- `prior_correction` Numeric value for legacy or low-level marginal-likelihood correction. Fixed-parameter prior removal is handled during AD objective construction.
- `fixed_prior_specs` Internal specification for removing fixed-parameter prior contributions during AD evaluation.
- `type` Character; the type of the model (e.g. 'lmer', 'glm', 'table', 'ttest', 'corr', 'mediation').
- `extra` List; used to store auxiliary data (like term assignments, factors, test results) needed by methods such as `classic()`.

## Methods

### Public methods:

- `RTMB_Model$new()`
- `RTMB_Model$prepare_init()`
- `RTMB_Model$get_par_list()`
- `RTMB_Model$get_ad_obj()`
- `RTMB_Model$to_unconstrained()`
- `RTMB_Model$to_constrained()`
- `RTMB_Model$unconstrained_vector_to_list()`
- `RTMB_Model$constrained_vector_to_list()`
- `RTMB_Model$calculate_satterthwaite_df()`
- `RTMB_Model$calculate_reml_satterthwaite_df()`
- `RTMB_Model$build_ad_obj()`
- `RTMB_Model$optimize()`
- `RTMB_Model$classic()`
- `RTMB_Model$sample()`
- `RTMB_Model$variational()`
- `RTMB_Model$print_code()`
- `RTMB_Model$print_log_prob()`
- `RTMB_Model$print_transform()`
- `RTMB_Model$print_generate()`
- `RTMB_Model$get_n_obs()`
- `RTMB_Model$fixed_model()`
- `RTMB_Model$clone()`

**Method** `new()`: Create a new 'RTMB\_Model' object.

#### Usage:

```
RTMB_Model$new(
  data,
  par_list,
  log_prob,
  transform = NULL,
  generate = NULL,
  par_names = NULL,
```

```

    init = NULL,
    view = NULL,
    code = NULL,
    gr_test = FALSE
  )

```

*Arguments:*

`data` Named list containing observation data.  
`par_list` Evaluated list of parameter definitions.  
`log_prob` Executable function for likelihood and prior calculations.  
`transform` Function for calculating transformed parameters (optional).  
`generate` Function for calculating generated quantities (optional).  
`par_names` List of variable names corresponding to parameter dimensions.  
`init` List or numeric vector of initial values.  
`view` Character vector of variable names to prioritize in summary.  
`code` Original model AST generated by `'rtmb_code()'`.  
`gr_test` Logical; if TRUE, evaluate the gradient once after the AD tape is built.

**Method** `prepare_init()`: Prepare and format initial values for the model parameters.

*Usage:*

```
RTMB_Model$prepare_init(init_arg)
```

*Arguments:*

`init_arg` Optional list or numeric vector of initial values. If NULL, defaults to `'self$init'` or random generation.

*Returns:* A flat numeric vector of constrained initial values.

**Method** `get_par_list()`: Get the current parameters as a named list.

*Usage:*

```
RTMB_Model$get_par_list(init = NULL)
```

*Arguments:*

`init` Optional numeric vector or list of initial values. If NULL, uses the model's current initial values.

*Returns:* A named list of constrained parameters.

**Method** `get_ad_obj()`: Get the RTMB automatic differentiation object.

*Usage:*

```
RTMB_Model$get_ad_obj(...)
```

*Arguments:*

`...` Arguments passed to `build_ad_obj`.

*Returns:* A TMB objective object (`ad_obj`).

**Method** `to_unconstrained()`: Convert a list of constrained parameters to unconstrained space.

*Usage:*

```
RTMB_Model$to_unconstrained(par_list_constrained)
```

*Arguments:*

`par_list_constrained` A named list of parameters in their natural space.

*Returns:* A named list of parameters in unconstrained space.

**Method** `to_constrained()`: Convert a list of unconstrained parameters to constrained (natural) space.

*Usage:*

```
RTMB_Model$to_constrained(par_list_unconstrained)
```

*Arguments:*

`par_list_unconstrained` A named list of parameters in unconstrained space.

*Returns:* A named list of parameters in constrained space.

**Method** `unconstrained_vector_to_list()`: Convert a flat unconstrained vector to a named list.

*Usage:*

```
RTMB_Model$unconstrained_vector_to_list(vec)
```

*Arguments:*

`vec` A numeric vector in unconstrained space.

*Returns:* A named list of unconstrained parameters.

**Method** `constrained_vector_to_list()`: Convert a flat constrained vector to a named list.

*Usage:*

```
RTMB_Model$constrained_vector_to_list(vec)
```

*Arguments:*

`vec` A numeric vector in constrained space.

*Returns:* A named list of constrained parameters.

**Method** `calculate_satterthwaite_df()`: Calculate Satterthwaite degrees of freedom for parameters.

*Usage:*

```
RTMB_Model$calculate_satterthwaite_df(
  ad_obj,
  idx_fix_active = NULL,
  L_u_total = NULL,
  opt_par = NULL,
  max_df = NULL,
  silent = FALSE,
  return_sensitivities = FALSE
)
```

*Arguments:*

`ad_obj` An RTMB objective object at the optimum.

`idx_fix_active` Integer vector; indices of active fixed parameters in the full unconstrained vector.

`L_u_total` Integer; total length of the full unconstrained parameter vector.

`opt_par` Optional numeric vector of optimized parameters.

`max_df` Numeric; maximum allowed degrees of freedom. Default is NULL.

`silent` Logical; whether to suppress informational messages. Default is FALSE.

`return_sensitivities` Logical; if TRUE, returns Hessian sensitivities and V for delta method.

*Returns:* A numeric vector of estimated degrees of freedom (length = `L_u_total`). Inf for random effects.

**Method** `calculate_reml_satterthwaite_df()`: Calculate Satterthwaite degrees of freedom for integrated fixed effects (REML).

*Usage:*

```
RTMB_Model$calculate_reml_satterthwaite_df(
  ad_obj,
  opt_par,
  beta_idx,
  max_df = NULL,
  silent = FALSE,
  return_sensitivities = FALSE
)
```

*Arguments:*

`ad_obj` An RTMB objective object.

`opt_par` Numeric vector of optimized variance components.

`beta_idx` Integer vector; indices of fixed effects within the random effects vector.

`max_df` Numeric; maximum allowed degrees of freedom. Default is NULL.

`silent` Logical; whether to suppress informational messages. Default is FALSE.

`return_sensitivities` Logical; if TRUE, returns Hessian sensitivities and delta-method components.

... Additional arguments.

*Returns:* A numeric vector of estimated degrees of freedom for the fixed effects.

**Method** `build_ad_obj()`: Build the RTMB automatic differentiation object.

*Usage:*

```
RTMB_Model$build_ad_obj(
  init = NULL,
  laplace = FALSE,
  jacobian_target = "none",
  map = NULL,
  .marginal_vars = NULL
)
```

*Arguments:*

`init` Optional numeric vector or list of initial values for the parameters. Default is NULL.

`laplace` Logical; whether to use Laplace approximation to marginalize random effects. Default is FALSE.

`jacobian_target` Character string specifying which parameters to apply Jacobian adjustments to:

- "all": Apply to all parameters (standard for Bayesian MCMC).
- "random": Apply only to parameters with `random = TRUE` (standard for Laplace approximation/REML).
- "none": No Jacobian adjustment (standard for Maximum Likelihood estimation).

Default is "none".

`map` Optional list specifying parameters to fix. Passed directly to `MakeADFun`. Default is `NULL`.

`.marginal_vars` Internal use for profiling marginalized parameters.

*Returns:* An RTMB objective object (`ad_obj`).

**Method** `optimize()`: Perform Maximum Likelihood or Maximum A Posteriori (MAP) estimation.

*Usage:*

```
RTMB_Model$optimize(
  laplace = TRUE,
  init = NULL,
  num_estimate = 1,
  control = list(),
  optimizer = "nlminb",
  method = "BFGS",
  map = NULL,
  fixed = NULL,
  se_method = c("wald", "sampling", "none"),
  num_samples = 1000,
  seed = 123,
  marginal = "none",
  df_method = "auto",
  view = NULL,
  ...
)
```

*Arguments:*

`laplace` Logical; whether to use Laplace approximation for random effects. Default is `TRUE`.

`init` Optional initial values for parameters (numeric vector or list).

`num_estimate` Integer; Number of multi-start optimization runs. Default is 1.

`control` A list of control settings passed to the optimizer.

`optimizer` Character; The optimizer to use: "nlminb" (default) or "optim".

`method` Character; The method for "optim" (e.g., "BFGS", "L-BFGS-B").

`map` Optional list specifying parameters to fix (factors).

`fixed` Optional list specifying parameter values to fix.

`se_method` Character; method for uncertainty estimation.

- "wald": Wald standard errors and Wald confidence intervals.
- "sampling": Simulation-based error propagation.
- "none": Do not compute standard errors, confidence intervals, or degrees of freedom.

`num_samples` Integer; number of samples to draw when `se_method` is "sampling". Default is 1000.

`seed` Integer; random seed for sampling.

`marginal` Character vector or "auto" or "none"; specifies which parameters to marginalize out via Laplace approximation.

- "none" (default): No additional marginalization.
- "auto": Uses variables specified by the model wrapper (e.g., fixed effects in lmer).
- "all": Marginalize all parameters (advanced use).
- character vector: Specific parameter names to marginalize.

`df_method` Character or numeric; method for finite degrees-of-freedom approximation. Used only when 'marginal' resolves to one or more parameters and 'se\_method' != "none".

- "auto": Uses Satterthwaite approximation when marginalization is active.
- "satterthwaite": Satterthwaite approximation.
- "inf" or "none": Use infinite degrees of freedom.
- numeric: Use the supplied fixed degrees of freedom.

`view` Character vector of parameter names to prioritize in summary display.

... Additional arguments.

*Returns:* A fitted 'MAP\_Fit' object.

**Method** `classic()`: Perform frequentist inference (REML/ML) with model-appropriate degrees of freedom.

*Usage:*

```
RTMB_Model$classic(
  df_method = "auto",
  se_method = c("wald", "sampling"),
  num_samples = 1000,
  seed = 123,
  view = NULL,
  map = NULL,
  fixed = NULL,
  ...
)
```

*Arguments:*

`df_method` Character; Method for calculating degrees of freedom. Default is "auto".

- "auto": Determined based on model type (e.g., Satterthwaite for mixed models).
- "residual": Residual degrees of freedom.
- "satterthwaite": Satterthwaite approximation (robust for mixed models).
- "inf": Use z-distribution (infinite degrees of freedom).

`se_method` Character; The method for CI estimation: "wald" (default) or "sampling".

`num_samples` Integer; number of samples to draw when `se_method` is "sampling". Default is 1000.

`seed` Integer; random seed for sampling.

`view` Character vector of parameters to prioritize in the summary output.

`map` Optional list specifying parameters to fix.

fixed Optional list specifying parameter values to fix.

... Additional arguments.

*Returns:* A 'Classic\_Fit' object.

**Method** sample(): Draw posterior samples from the model.

*Usage:*

```
RTMB_Model$sample(
  sampling = 1000,
  warmup = 1000,
  chains = 4,
  thin = 1,
  seed = sample.int(1e+06, 1),
  delta = 0.8,
  max_treedepth = 10,
  nuts_variant = c("multinomial", "slice"),
  metric = c("auto", "diag", "dense", "hybrid"),
  metric_init = c("identity", "hessian"),
  metric_adaptation = c("stan_window", "cumulative", "window"),
  metric_regularization = TRUE,
  metric_shrinkage = 5,
  metric_min = 1e-06,
  metric_max = 1e+06,
  parallel = FALSE,
  laplace = FALSE,
  init = NULL,
  init_jitter = 0.1,
  save_csv = NULL,
  map = NULL,
  fixed = NULL,
  globals = FALSE,
  progress = c("auto", "none", "bar", "message"),
  .resume = NULL
)
```

*Arguments:*

sampling Number of sampling iterations. Default is 1000.

warmup Number of warmup iterations. Default is 1000.

chains Number of MCMC chains. Default is 4.

thin Thinning interval. Default is 1.

seed Random seed.

delta Target acceptance rate for HMC/NUTS. Default is 0.8.

max\_treedepth Maximum tree depth for HMC/NUTS. Default is 10.

nuts\_variant NUTS proposal selection variant. "multinomial" uses Stan-style Hamiltonian-weighted proposal selection; "slice" uses the original slice-based implementation. Default is "multinomial".

metric Mass matrix adaptation type. "auto" starts from the hybrid strategy and falls back to "diag" during warmup when the dense covariance block is unstable or has very low correlation; "diag" adapts only marginal variances; "dense" adapts a full covariance metric for

correlated parameters; "hybrid" uses a dense metric for fixed parameters and a diagonal metric for random parameters. Default is "auto".

`metric_init` Initial metric source. "identity" starts from the unit metric; "hessian" starts from an inverse Hessian approximation at the chain initial value when available. Default is "identity".

`metric_adaptation` Metric learning mode during warmup. "stan\_window" uses the current expanding Stan-style slow window; "cumulative" uses all slow-window warmup draws accumulated so far; "window" uses only the current adaptation window. Default is "stan\_window".

`metric_regularization` Logical; whether to shrink the adapted metric toward the unit metric during warmup. Default is TRUE.

`metric_shrinkage` Non-negative scalar controlling the strength of metric shrinkage. Larger values shrink more strongly toward 1. Default is 5.

`metric_min` Minimum allowed diagonal metric value. Default is 1e-6.

`metric_max` Maximum allowed diagonal metric value. Default is 1e6.

`parallel` Logical; whether to run chains in parallel. Default is FALSE.

`laplace` Logical; whether to use Laplace approximation. Default is FALSE.

`init` Optional initial values for parameters.

`init_jitter` sd of randomize initial values for parameters.

`save_csv` Optional list for saving MCMC results. e.g., `list(name = "model", dir = "BayesRTMB_mcmc")`.

`map` Optional list specifying parameters to fix (factors).

`fixed` Optional list specifying parameter values to fix.

`globals` Logical; if TRUE, let **future** discover globals automatically when `parallel = TRUE`. If FALSE, only the model object components required by BayesRTMB workers are exported. Default is FALSE.

`progress` Progress reporting style. "auto", "bar", and "message" use line-based progress messages; "none" suppresses detailed progress but still prints the high-level start message unless `silent = TRUE` was used when creating the model. "bar" is accepted for backward compatibility. Default is "auto".

`.resume` Internal saved sampler state used by `'MCMC_Fit$continue_sampling()'`; users should leave this as 'NULL'.

*Returns:* A fitted 'MCMC\_Fit' object.

**Method** `variational()`: Run Automatic Differentiation Variational Inference (ADVI).

*Usage:*

```
RTMB_Model$variational(
  iter = 3000,
  tol_rel_obj = 0.005,
  window_size = 100,
  num_samples = 1000,
  num_estimate = 4,
  alpha = 0.01,
  laplace = FALSE,
  print_freq = 1000,
  method = c("meanfield", "fullrank", "hybrid"),
  parallel = FALSE,
```

```

    seed = sample.int(1e+06, 1),
    init = NULL,
    save_csv = NULL,
    map = NULL,
    fixed = NULL,
    globals = FALSE,
    progress = c("auto", "none", "bar", "message")
)

```

*Arguments:*

**iter** Integer; fixed number of iterations for the optimization. Default is 3000.

**tol\_rel\_obj** Numeric; relative tolerance for the ELBO change to determine convergence. Default is 0.005.

**window\_size** Integer; window size for median smoothing in the convergence check. Default is 100.

**num\_samples** Integer; number of posterior samples to generate from the fitted variational distribution. Default is 1000.

**num\_estimate** Integer; number of times to run the VB estimation (treated as chains). Default is 4.

**alpha** Numeric; learning rate for the Adam optimizer. Default is 0.01.

**laplace** Logical; whether to use Laplace approximation to marginalize random effects. Default is FALSE.

**print\_freq** Integer; iterations interval for progress output. Set to 0 to disable. Default is 1000.

**method** Character; method of Variational Inference. Default is "meanfield".

**parallel** Logical; whether to run estimations in parallel. Default is FALSE.

**seed** Integer; random seed for reproducibility.

**init** Optional numeric vector or list for initial parameter values. Default is NULL.

**save\_csv** Optional list for saving VB results. e.g., list(name = "model", dir = "BayesRTMB\_vb").

**map** Optional list specifying parameters to fix (factors).

**fixed** Optional list specifying parameter values to fix.

**globals** Logical; if TRUE, let **future** discover globals automatically when parallel = TRUE. If FALSE, only the model object components required by BayesRTMB workers are exported. Default is FALSE.

**progress** Progress reporting style. "auto" and "bar" use line-based progress messages; "none" suppresses detailed progress but still prints the high-level start message unless silent = TRUE was used when creating the model. Default is "auto".

*Returns:* A fitted 'VB\_Fit' object containing posterior samples and diagnostic information.

**Method print\_code():** Print model code or model structure.

*Usage:*

```
RTMB_Model$print_code()
```

*Returns:* The object itself, invisibly.

**Method print\_log\_prob():** Print the internal log-probability function.

*Usage:*

```
RTMB_Model$print_log_prob()
```

*Returns:* The object itself, invisibly.

**Method** `print_transform()`: Print the internal transformation function.

*Usage:*

```
RTMB_Model$print_transform()
```

*Returns:* The object itself, invisibly.

**Method** `print_generate()`: Print the internal generation function.

*Usage:*

```
RTMB_Model$print_generate()
```

*Returns:* The object itself, invisibly.

**Method** `get_n_obs()`: Automatically detect and count the number of independent data points (observations).

*Usage:*

```
RTMB_Model$get_n_obs()
```

*Returns:* Integer; total number of observations, or NULL if model code is missing.

**Method** `fixed_model()`: Create a model with fixed parameters.

*Usage:*

```
RTMB_Model$fixed_model(fixed = list(), silent = FALSE)
```

*Arguments:*

`fixed` A named list of parameter values to fix.

`silent` Logical; whether to suppress informational messages. Default is FALSE.

*Returns:* A new RTMB\_Model object.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
RTMB_Model$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Description

Models defined in `rtmb_code` rely on Automatic Differentiation (AD) via the RTMB package. To ensure the model is differentiable and numerically stable, specific coding practices must be followed.

## Details

**1. Automatic Differentiation and advector:** Parameters and intermediate calculations involving them are treated as `advector` objects. RTMB "records" all mathematical operations to compute derivatives automatically. If a function strips these attributes or is not differentiable, the gradient calculation will fail.

**2. Avoid Discrete Branching:** Standard R conditional statements like `if (x > 0)` or `ifelse()` based on parameter values do not provide derivatives.

- **Problem:** They create "jumps" in the likelihood surface.
- **Solution:** Use smooth approximations. For example, use `fabs` instead of `abs`, or `log_mix` for mixture logic.

**3. Numerical Stability:** Computations in log-space are preferred to prevent overflow or underflow. Use the following stable utilities instead of raw algebraic expressions:

- `log_sum_exp`: For summing probabilities in log-space.
- `log1p_exp`: For  $\log(1 + \exp(x))$ .
- `inv_logit`: For mapping real numbers to probabilities.

**4. Vectorization vs. Loops:**

- **Vectorization:** Highly recommended for performance. Standard R vectorized arithmetic (`+`, `-`, `*`, `/`, `log`, `exp`) works seamlessly with AD.
- **Loops:** Standard for loops are safe as long as the operations inside are differentiable.
- **Avoid apply:** Functions like `apply`, `sapply`, or `lapply` may sometimes strip AD attributes and should be replaced with vectorized operations or explicit loops.

**5. Matrix Operations:** Use specialized functions for matrix algebra to maintain efficiency:

- `quad_form_chol`: For efficient quadratic forms.
- `log_det_chol`: For log-determinants via Cholesky factors.

## See Also

[math\\_functions](#), [distributions](#), [rtmb\\_code](#)

rtmb\_table

*RTMB-based Contingency Table Analysis (Chi-squared Test)***Description**

‘rtmb\_table’ performs a chi-squared test of independence between two categorical variables. It provides both classic (frequentist) Pearson chi-squared tests and Bayesian multinomial-style models.

**Usage**

```
rtmb_table(
  x,
  y = NULL,
  data = NULL,
  correct = TRUE,
  prior = prior_flat(),
  fixed = NULL,
  WAIC = FALSE,
  ...
)
```

**Arguments**

|         |                                                                                                                                                                                               |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x       | Variable name, formula, table, or matrix.                                                                                                                                                     |
| y       | Variable name (optional if x is a formula).                                                                                                                                                   |
| data    | A data frame.                                                                                                                                                                                 |
| correct | Logical; if TRUE, apply Yates’ continuity correction for 2x2 classic analyses.                                                                                                                |
| prior   | Prior specification (Bayesian mode). Use ‘prior_flat()’ for a uniform Dirichlet prior or ‘prior_normal(dirichlet_alpha = ...)’ to set the Dirichlet concentration. Default is ‘prior_flat()’. |
| fixed   | Optional named list of fixed values for specific parameters.                                                                                                                                  |
| WAIC    | Logical; if TRUE, add pointwise ‘log_lik’ to the generate block for WAIC.                                                                                                                     |
| ...     | Reserved; unused arguments are rejected.                                                                                                                                                      |

**Value**

An ‘RTMB\_Model’ object.

**Examples**

```
# Classic chi-squared test
data(debate, package = "BayesRTMB")
rtmb_table(skill, cond, data = debate)$classic()
rtmb_table(table(debate$skill, debate$cond))$classic()
```

---

rtmb\_ttest

*RTMB-based Bayesian two-sample t-test wrapper function*


---

## Description

Performs a Bayesian or Frequentist two-sample t-test using RTMB.

## Usage

```
rtmb_ttest(
  x,
  y = NULL,
  data = NULL,
  r = 0.707,
  paired = FALSE,
  ID = NULL,
  y_range = NULL,
  prior = prior_flat(),
  init = NULL,
  fixed = NULL,
  var.equal = TRUE,
  missing = c("listwise", "fiml"),
  WAIC = FALSE,
  ...
)
```

## Arguments

|         |                                                                                                                                                                                                                                                        |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x       | Numeric vector of responses for group 1, a formula (e.g., 'y ~ group'), or a column name (unquoted) if 'data' is provided.                                                                                                                             |
| y       | Numeric vector of responses for group 2, or a column name (unquoted) if 'data' is provided. Required if 'x' is not a formula.                                                                                                                          |
| data    | Data frame containing the variables.                                                                                                                                                                                                                   |
| r       | Numeric; Cauchy prior scale for the effect size (delta). Default is 0.707.                                                                                                                                                                             |
| paired  | Logical; whether to perform a paired t-test.                                                                                                                                                                                                           |
| ID      | Character; name of the ID variable for paired t-tests (required for formula input with paired = TRUE).                                                                                                                                                 |
| y_range | Theoretical minimum and maximum values of the response variable as a vector c(min, max). Required when using weakly informative priors.                                                                                                                |
| prior   | An object of class "rtmb_prior". Use 'prior_flat()' for no prior, 'prior_normal()' for default normal/exponential priors, 'prior_jzs()' for JZS t-test priors, or 'prior_weak()' for weakly informative Bayesian inference. Default is 'prior_flat()'. |
| init    | List of initial values.                                                                                                                                                                                                                                |
| fixed   | Optional named list of fixed values for specific parameters.                                                                                                                                                                                           |

|                        |                                                                           |
|------------------------|---------------------------------------------------------------------------|
| <code>var.equal</code> | Logical; whether to assume equal variances. Default is TRUE.              |
| <code>missing</code>   | Missing value handling strategy: "listwise".                              |
| <code>WAIC</code>      | Logical; if TRUE, add pointwise 'log_lik' to the generate block for WAIC. |
| <code>...</code>       | Reserved; unused arguments are rejected.                                  |

## Details

For classic inference, heteroscedastic two-sample t-tests use the same RTMB Satterthwaite machinery as `optimize(marginal = ..., df_method = "satterthwaite")`. The result is model-based and reproducible from the printed model code. This corresponds to the Welch-type unequal-variance t-test, but the degrees of freedom are computed by the package's internal Satterthwaite procedure rather than by a separate closed-form formula. For JZS t-tests, `prior_jzs()` places a Cauchy prior on the standardized effect size and the Jeffreys scale prior  $p(\sigma) \propto 1/\sigma$  on the residual standard deviation. For paired tests, the latter is applied to the standard deviation of the pairwise differences. When `prior_jzs()` is combined with `var.equal = FALSE`, BayesRTMB uses a Welch-style JZS extension: the effect size 'delta' is an explicit parameter with a Cauchy prior, and the group mean difference is scaled by the root-mean-square of the two group standard deviations. The Jeffreys scale prior is applied separately to both group standard deviations.

## Value

An 'RTMB\_Model' object.

## Examples

```
# Simulate two-sample data with a true effect size
set.seed(123)
y1 <- rnorm(30, mean = 0.5, sd = 1)
y2 <- rnorm(30, mean = 0.0, sd = 1)

# Fit the Bayesian two-sample t-test model
# r = 0.707 is the standard scale for the Cauchy prior on the effect size
fit_ttest <- rtmb_ttest(y1, y2, prior = prior_jzs(r = 0.707))

# MCMC sampling (chains and iterations reduced for faster execution)
mcmc_ttest <- fit_ttest$sample(sampling = 500, warmup = 500, chains = 2)
mcmc_ttest$summary()
```

---

rtmb\_vector

---

Create an AD-compatible vector

---

## Description

Creates a fixed-length vector that is safe to use inside `rtmb_code()` when subsequent assignments may involve RTMB automatic-differentiation values. This is a safer alternative to `numeric(length)` in model code.

**Usage**

```
rtmb_vector(value = 0, length, seed = NULL)
```

**Arguments**

|        |                                                                                                                                                                                                 |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| value  | Initial value. Usually a scalar such as '0'.                                                                                                                                                    |
| length | Length of the vector.                                                                                                                                                                           |
| seed   | Optional AD value used to create the container. Usually this can be left as 'NULL'; inside 'rtmb_code()', BayesRTMB automatically provides an AD seed from the model parameters when available. |

**Value**

An AD-compatible vector.

---

rtmb\_wrappers

---

*Common Features and Arguments of RTMB Wrapper Functions*


---

**Description**

The RTMB wrapper functions ('rtmb\_lm', 'rtmb\_glm', 'rtmb\_glmer', 'rtmb\_fa', etc.) share a unified interface designed to make Bayesian and Frequentist inference accessible through familiar R formulas and standard model specifications.

**Details**

All wrapper functions in this package are built upon the same core engine. This ensures that regardless of the model type, the workflow for estimation, summary, and expansion remains consistent.

**1. Unified Inference Methods:** Every model object returned by a wrapper function provides the following methods:

- `$classic()`: Performs standard frequentist estimation (MLE/REML) using 'prior\_flat()'. Supports robust standard errors and Satterthwaite/Between-Within degrees of freedom approximations.
- `$optimize()`: Performs Maximum A Posteriori (MAP) estimation (Empirical Bayes).
- `$sample()`: Performs MCMC sampling (NUTS via tmbstan) for full Bayesian inference.
- `$variational()`: Performs Variational Inference (ADVI) for fast posterior approximation.

**2. Prior API and Regularization:** You can specify the prior distribution using the 'prior' argument:

- `prior_flat()`: No additional prior (default). Required for `classic()` inference.
- `prior_normal()`: Manual normal/exponential priors for parameters.
- `prior_weak()`: Weakly informative priors based on `y_range`.
- `prior_rhs()`: Regularized Horseshoe prior for continuous shrinkage.

- `prior_ssp()`: Spike-and-Slab prior for sparse variable selection.

*Note: When using 'prior\_weak()', 'prior\_rhs()', or 'prior\_ssp()', you must specify `y_range = c(min, max)` to let the model set appropriate global scales for the priors.*

**3. Model Comparison and Bayes Factors:** Model comparison is performed exclusively via Bridge Sampling. To compare a full model against a nested null model, use the `$bayes_factor(fixed = list(...))` method on an MCMC fit object. For example, `fit$bayes_factor(fixed = list(b = 0))` tests the hypothesis that the fixed effect `b` is exactly zero.

**4. Standard Errors and Degrees of Freedom:** The `$classic()` and `$optimize()` methods provide advanced summary options:

- `se_method`: Calculate "robust" (Huber-White) or "cluster-robust" standard errors by providing a cluster variable.
- `df_method`: Use "satterthwaite" for mixed models or "bw" (Between-Within) for multi-level correlation models to obtain accurate p-values and confidence intervals.

**5. Fixed parameters:** Use `fixed = list(...)` during model construction to fix model parameters to specified values. For example, `fixed = list(delta = 0)` fixes the parameter `delta` to zero. This same syntax is used in `$bayes_factor()` to define the restricted model.

---

|       |                                       |
|-------|---------------------------------------|
| r_hat | Calculate Rank-normalized Split-R-hat |
|-------|---------------------------------------|

---

## Description

Calculate Rank-normalized Split-R-hat

## Usage

```
r_hat(sims)
```

## Arguments

sims                      A matrix of samples (iterations x chains).

## Value

A numeric value.

---

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| safe_rtmb_model | <i>Safe RTMB model construction (with error message translation)</i> |
|-----------------|----------------------------------------------------------------------|

---

**Description**

Wraps ‘rtmb\_model’ to translate cryptic error messages generated during ‘MakeADFun’ execution (originating from C++/RTMB) into user-friendly hints.

**Usage**

```
safe_rtmb_model(data, code, par_names = list(), init = NULL, view = NULL)
```

**Arguments**

|           |                                                                                                   |
|-----------|---------------------------------------------------------------------------------------------------|
| data      | A list of data that has passed ‘validate_data’.                                                   |
| code      | Code blocks for likelihood and priors defined with ‘model_code()’.                                |
| par_names | A list of variable names corresponding to the dimensions of each parameter (optional).            |
| init      | A list or numeric vector of initial values (optional).                                            |
| view      | Character vector of parameter names to be displayed preferentially in summary outputs (optional). |

**Value**

Returns an ‘rtmb\_model’ object (R6 class instance) upon successful compilation.

---

|                |                                 |
|----------------|---------------------------------|
| simple_effects | <i>Calculate Simple Effects</i> |
|----------------|---------------------------------|

---

**Description**

Calculate the effect of a focal variable at different levels of a moderator. For categorical focal variables, it calculates pairwise differences (contrasts). For continuous focal variables, it calculates the slope (simple slopes).

**Usage**

```
simple_effects(
  fit,
  effect,
  prob = 0.95,
  sd_multiplier = 1,
  sd_slice = NULL,
  ...
)
```

**Arguments**

|                            |                                                                                                                                                                                                                                                        |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>fit</code>           | Model fit object (e.g., 'map_fit', 'mcmc_fit').                                                                                                                                                                                                        |
| <code>effect</code>        | Character string of the interaction (e.g., "A:B"). The first variable is the focal variable.                                                                                                                                                           |
| <code>prob</code>          | Probability for the credible/confidence interval (default is 0.95).                                                                                                                                                                                    |
| <code>sd_multiplier</code> | Multiplier for SD for continuous moderators (default is 1).                                                                                                                                                                                            |
| <code>sd_slice</code>      | Logical or NULL. If TRUE, continuous moderators are evaluated at mean - SD, mean, and mean + SD. If FALSE, all observed moderator values are used. If NULL (default), sd slicing is used automatically when the moderator has 6 or more unique values. |
| <code>...</code>           | Additional arguments.                                                                                                                                                                                                                                  |

**Value**

A 'ce\_simple' object (data frame) containing the estimated effects and their credible intervals. For 'Classic\_Fit' objects, test columns ('df', 't value', and 'Pr') are also returned when standard errors are available.

**Examples**

```
data(debate, package = "BayesRTMB")
fit <- rtmb_lm(sat ~ talk * perf, data = debate)
map_fit <- fit$optimize()
# Effect of talk at different levels of performance
se <- simple_effects(map_fit, effect = "talk:perf")
print(se)
```

---

```
simple_effects.mcmc_fit
```

*Simple effects for MCMC fit objects*

---

**Description**

Simple effects for MCMC fit objects

**Usage**

```
## S3 method for class 'mcmc_fit'
simple_effects(
  fit,
  effect,
  prob = 0.95,
  sd_multiplier = 1,
  sd_slice = NULL,
  ...
)
```

**Arguments**

|               |                                                                                                          |
|---------------|----------------------------------------------------------------------------------------------------------|
| fit           | An object of class 'MCMC_Fit'.                                                                           |
| effect        | Interaction term (e.g., "A:B").                                                                          |
| prob          | Probability for credible intervals.                                                                      |
| sd_multiplier | Multiplier for SD for continuous moderators.                                                             |
| sd_slice      | Logical or NULL; controls whether continuous moderators are evaluated at mean - SD, mean, and mean + SD. |
| ...           | Additional arguments.                                                                                    |

---

|         |                         |
|---------|-------------------------|
| softmax | <i>Softmax function</i> |
|---------|-------------------------|

---

**Description**

Softmax function

**Usage**

```
softmax(x)
```

**Arguments**

|   |                   |
|---|-------------------|
| x | A numeric vector. |
|---|-------------------|

**Value**

A numeric vector of softmax probabilities.

---

|               |                                                |
|---------------|------------------------------------------------|
| sort_loadings | <i>Sort and display factor loadings neatly</i> |
|---------------|------------------------------------------------|

---

**Description**

Sort and display factor loadings neatly

**Usage**

```
sort_loadings(loadings, cutoff = 0, round_digits = 3)
```

**Arguments**

|              |                                                                                     |
|--------------|-------------------------------------------------------------------------------------|
| loadings     | Matrix, data frame, or list of factor loadings (if list, the first element is used) |
| cutoff       | Absolute loadings below this value will be displayed as blank (default is 0.0)      |
| round_digits | Number of decimal places to display (default is 3)                                  |

**Value**

Sorted loading matrix (returned invisibly, allowing assignment to a variable)

---

|                  |                                   |
|------------------|-----------------------------------|
| squared_distance | <i>Squared Euclidean distance</i> |
|------------------|-----------------------------------|

---

**Description**

Squared Euclidean distance

**Usage**

```
squared_distance(x, y, eps = 1e-08)
```

**Arguments**

- x                    A numeric vector.
- y                    A numeric vector.
- eps                  A small value added for numerical stability (default is 1e-8).

**Value**

The squared Euclidean distance between x and y.

---

|           |                           |
|-----------|---------------------------|
| stz_basis | <i>stz basis function</i> |
|-----------|---------------------------|

---

**Description**

stz basis function

**Usage**

```
stz_basis(K)
```

**Arguments**

- K                    A numeric value

**Value**

A transformation matrix

---

|                 |                                         |
|-----------------|-----------------------------------------|
| summary.ce_rtmb | <i>Summary method for ce_rtmb class</i> |
|-----------------|-----------------------------------------|

---

**Description**

Summary method for ce\_rtmb class

**Usage**

```
## S3 method for class 'ce_rtmb'
summary(object, ...)
```

**Arguments**

|        |                            |
|--------|----------------------------|
| object | An object of class ce_rtmb |
| ...    | Additional arguments.      |

---

|              |                                                |
|--------------|------------------------------------------------|
| summary_mcmc | <i>Summarize MCMC Draws Stored in an Array</i> |
|--------------|------------------------------------------------|

---

**Description**

Summarizes posterior draws stored in the same array layout used by [MCMC\\_Fit](#): iterations by chains by variables. The returned object uses the same columns and print method as `MCMC_Fit$summary()`.

**Usage**

```
summary_mcmc(draws, pars = NULL, chains = NULL, max_rows = 10, digits = 2)
```

**Arguments**

|          |                                                                                                                                                                                                                      |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| draws    | A numeric three-dimensional array with dimensions iterations by chains by variables.                                                                                                                                 |
| pars     | Optional numeric or character vector selecting variables. Character values may be full variable names (for example, "beta[1]") or base names (for example, "beta"). Prefix character names with "-" to exclude them. |
| chains   | Optional numeric vector selecting chains. Positive and negative integer indexing are supported.                                                                                                                      |
| max_rows | Maximum number of variables to include. Use 'NULL' to include all variables.                                                                                                                                         |
| digits   | Number of decimal places used when printing the result.                                                                                                                                                              |

**Value**

A data frame with class "summary\_BayesRTMB" containing posterior means, standard deviations, marginal MAP estimates, 95 percent intervals, bulk and tail effective sample sizes, and split R-hat values.

Examples

```
set.seed(123)
draws <- array(
  rnorm(200 * 2 * 2),
  dim = c(200, 2, 2),
  dimnames = list(NULL, c("chain1", "chain2"), c("alpha", "beta"))
)
summary_mcmc(draws)
```

---

|             |                                   |
|-------------|-----------------------------------|
| sum_to_zero | <i>Sum-to-zero transformation</i> |
|-------------|-----------------------------------|

---

Description

Transforms a vector of length K-1 to a vector of length K that sums to zero using an orthogonal basis matrix.

Usage

```
sum_to_zero(x)
```

Arguments

x                      A numeric vector of length K-1.

Value

A numeric vector of length K whose elements sum to zero.

---

|           |                                            |
|-----------|--------------------------------------------|
| test_info | <i>Calculate Test Information Function</i> |
|-----------|--------------------------------------------|

---

Description

Calculate Test Information Function

Usage

```
test_info(x, ...)
```

Arguments

x                      An object of class RTMB\_Fit\_Base  
...                    Additional arguments.

**Examples**

```
fit <- rtmb_irt(data = BigFive[, 1:5], model = "2PL", type = "ordered")
map_fit <- fit$optimize()
ti <- test_info(map_fit)
plot(ti)
```

---

|                    |                                                    |
|--------------------|----------------------------------------------------|
| to_centered_matrix | <i>Vector to centered matrix (RTMB compatible)</i> |
|--------------------|----------------------------------------------------|

---

**Description**

Vector to centered matrix (RTMB compatible)

**Usage**

```
to_centered_matrix(x, R, C)
```

**Arguments**

|   |                                       |
|---|---------------------------------------|
| x | A numeric vector of length (R-1) * C. |
| R | Number of rows.                       |
| C | Number of columns.                    |

**Value**

An R x C matrix where each column sums to zero.

---

|                 |                                                               |
|-----------------|---------------------------------------------------------------|
| to_centered_tri | <i>Vector to centered triangular matrix (RTMB compatible)</i> |
|-----------------|---------------------------------------------------------------|

---

**Description**

Vector to centered triangular matrix (RTMB compatible)

**Usage**

```
to_centered_tri(x, R, C)
```

**Arguments**

|   |                                         |
|---|-----------------------------------------|
| x | A numeric vector of appropriate length. |
| R | Number of rows.                         |
| C | Number of columns.                      |

**Value**

An R x C matrix with column-wise sum-to-zero constraints on lower elements.

---

|         |                                         |
|---------|-----------------------------------------|
| to_long | <i>Convert Wide Data to Long Format</i> |
|---------|-----------------------------------------|

---

**Description**

A highly intuitive wrapper around `stats::reshape` designed for psychological research. It converts data from wide format to long format by identifying within-subjects factors.

**Usage**

```
to_long(  
  data,  
  within = NULL,  
  label = NULL,  
  value = "Value",  
  id = NULL,  
  sort = FALSE  
)
```

**Arguments**

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data   | A data frame in wide format.                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| within | The columns to gather into long format. Can be: <ul style="list-style-type: none"><li>• A character vector of column names.</li><li>• A range string like "time1:time4".</li><li>• A prefix string like "time" (matches all columns starting with "time").</li><li>• A list of column-name vectors or range/prefix specifications to concatenate into one long value column.</li><li>• Multiple range/prefix/name specifications when value has the same length.</li></ul> |
| label  | The name for the new column that will contain the level names (e.g., "Time"). If NULL, it defaults to the prefix used in within or "Condition".                                                                                                                                                                                                                                                                                                                            |
| value  | The name for the new column that will contain the measurement values. Can be a character vector for multiple measurement groups. Default is "Value".                                                                                                                                                                                                                                                                                                                       |
| id     | The identifier columns that should be repeated for each row. If NULL (default), all columns NOT specified in within are treated as IDs.                                                                                                                                                                                                                                                                                                                                    |
| sort   | Logical; if TRUE, sort the output by id and the within label. If FALSE (default), preserve the original row order of data and the within-column order inside each row.                                                                                                                                                                                                                                                                                                     |

**Value**

A data frame in long format.

---

|              |                                                            |
|--------------|------------------------------------------------------------|
| to_lower_tri | <i>Vector to lower triangular matrix (RTMB compatible)</i> |
|--------------|------------------------------------------------------------|

---

**Description**

Vector to lower triangular matrix (RTMB compatible)

**Usage**

```
to_lower_tri(x, M, D)
```

**Arguments**

|   |                        |
|---|------------------------|
| x | A numeric or advector. |
| M | Number of rows.        |
| D | Number of columns.     |

**Value**

An M x D lower triangular matrix.

---

|         |                                         |
|---------|-----------------------------------------|
| to_wide | <i>Convert Long Data to Wide Format</i> |
|---------|-----------------------------------------|

---

**Description**

A user-friendly wrapper around `stats::reshape` to convert data from long format back to wide format.

**Usage**

```
to_wide(data, within = "Condition", value = "Value", id = NULL)
```

**Arguments**

|        |                                                                                                                                   |
|--------|-----------------------------------------------------------------------------------------------------------------------------------|
| data   | A data frame in long format.                                                                                                      |
| within | The name of the column containing within-subjects factor levels (e.g., "Condition").                                              |
| value  | The name of the column containing measurement values (e.g., "Value"). Can be a character vector for multiple measurement columns. |
| id     | The identifier columns (e.g., "Subject", "Group"). If NULL, inferred as all columns except within and value.                      |

**Value**

A data frame in wide format.

---

|          |                                    |
|----------|------------------------------------|
| training | <i>Social Skills Training Data</i> |
|----------|------------------------------------|

---

### Description

A small dataset containing repeated outcome measurements from a social skills training study. The outcome was measured at four time points for participants assigned to a control or intervention condition.

### Usage

```
training
```

### Format

A data frame with 12 rows and 8 columns:

**ID** Participant identifier.

**time1** Outcome score at the first measurement occasion.

**time2** Outcome score at the second measurement occasion.

**time3** Outcome score at the third measurement occasion.

**time4** Outcome score at the fourth measurement occasion.

**a** Training condition indicator: 0 = control group, 1 = intervention group.

**b** Qualitative baseline skill classification.

**age** Participant age.

### Source

Simulated dummy data created by the package author.

---

|                |                                          |
|----------------|------------------------------------------|
| transform_code | <i>Transformed Code Wrapper for RTMB</i> |
|----------------|------------------------------------------|

---

### Description

Transformed Code Wrapper for RTMB

### Usage

```
transform_code(expr, env = parent.frame(), ad_seed_name = NULL)
```

**Arguments**

|              |                                                                     |
|--------------|---------------------------------------------------------------------|
| expr         | A block of code containing calculations for transformed parameters. |
| env          | Environment to assign to the generated function.                    |
| ad_seed_name | Optional parameter name used internally to seed RTMB's AD type.     |

**Value**

A function taking (dat, par) that returns a named list.

---

|             |                                                                              |
|-------------|------------------------------------------------------------------------------|
| upgrade_fit | <i>Upgrade a saved fit object to the current BayesRTMB class definitions</i> |
|-------------|------------------------------------------------------------------------------|

---

**Description**

'upgrade\_fit()' rebuilds an existing fit object with the currently loaded BayesRTMB R6 class definitions. This is useful for saved fit objects created by older package versions when newer methods have been added to the fit classes.

**Usage**

```
upgrade_fit(fit, upgrade_model = TRUE)
```

**Arguments**

|               |                                                                                                                                                                                           |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fit           | A BayesRTMB fit object. Supported classes are MCMC, VB, MAP, and classic fit objects.                                                                                                     |
| upgrade_model | Logical; if 'TRUE', rebuild the embedded model object from its stored data, parameter list, and model functions. If rebuilding fails, the original model is kept and a warning is issued. |

**Details**

The function does not refit the model or recompute estimates. It only copies the stored estimation results into a new fit object. When 'upgrade\_model' is 'TRUE', the embedded 'RTMB\_Model' object is also rebuilt from the model components stored in the fit, so newer model methods are available as well.

**Value**

A fit object of the same type, using the currently loaded class definitions.

---

|               |                                              |
|---------------|----------------------------------------------|
| validate_data | <i>Pre-validation of data and parameters</i> |
|---------------|----------------------------------------------|

---

### Description

Before passing data to ‘rtmb\_model’, it checks for the presence of R-specific data types (such as `data.frame`) and missing values, and outputs errors or warnings accordingly.

### Usage

```
validate_data(dat_list)
```

### Arguments

|                       |                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------|
| <code>dat_list</code> | A list of data to be passed to the model (usually containing matrices or numeric vectors). |
|-----------------------|--------------------------------------------------------------------------------------------|

### Details

RTMB’s automatic differentiation engine requires pure ‘matrix’ or ‘numeric’ types. If a user mistakenly passes a ‘data.frame’, incomprehensible errors occur during the construction of the computation graph. This function catches such issues in advance.

### Value

Returns invisible ‘NULL’ on success. Interrupts execution with ‘stop()’ or issues ‘warning()’ if issues are found.

---

|        |                      |
|--------|----------------------|
| VB_Fit | <i>VB fit object</i> |
|--------|----------------------|

---

### Description

VB fit object

VB fit object

### Details

An R6 class storing posterior samples and related information from Automatic Differentiation Variational Inference (ADVI).

### Super class

`BayesRTMB::RTMB_Fit_Base -> advi_fit`

**Public fields**

`model` An 'RTMB\_Model' object used for estimation.

`fit` A 3D array of posterior draws for fixed model parameters.

`random_fit` A 3D array of posterior draws for random effects, if available.

`transform_fit` A 3D array of posterior draws for transformed parameters, if available.

`generate_fit` A 3D array of posterior draws for generated quantities, if available.

`transform_dims` A list storing dimension information for transformed parameters.

`generate_dims` A list storing dimension information for generated quantities.

`elbo_history` A list of numeric vectors storing the Evidence Lower Bound (ELBO) history during optimization for each chain.

`laplace` Logical; whether Laplace approximation was used to marginalize random effects.

`posterior_mean` A named numeric vector of posterior mean estimates.

`ELBO` A numeric vector of final ELBO values for each chain.

`rel_obj_vals` A numeric vector of final relative objective tolerance values for each chain.

`best_chain` Integer; the index of the chain with the maximum ELBO.

`mu_history` Matrix of the parameter trajectory from the final window.

**Methods****Public methods:**

- `VB_Fit$get_point_estimate()`
- `VB_Fit$estimate()`
- `VB_Fit$EAP()`
- `VB_Fit$MAP()`
- `VB_Fit$new()`
- `VB_Fit$print()`
- `VB_Fit$draws()`
- `VB_Fit$summary()`
- `VB_Fit$plot_elbo()`
- `VB_Fit$plot_trajectory()`
- `VB_Fit$transformed_draws()`
- `VB_Fit$generated_quantities()`
- `VB_Fit$WAIC()`
- `VB_Fit$diagnose()`
- `VB_Fit$clone()`

**Method** `get_point_estimate()`: Get point estimate for a target parameter.

*Usage:*

```
VB_Fit$get_point_estimate(target, chains = NULL, best_chains = NULL)
```

*Arguments:*

`target` Target parameter name.

chains Numeric vector of chains to include. If NULL, the best chain is used.  
 best\_chains Integer; number of best chains to retain based on ELBO.

*Returns:* Matrix, array, vector, or scalar point estimate.

**Method** estimate(): Get point estimates from variational draws.

*Usage:*

```
VB_Fit$estimate(
  pars = NULL,
  type = c("mean", "EAP", "marginal_map", "joint_map", "MAP"),
  component = c("all", "parameters", "transform", "generate"),
  chains = NULL,
  best_chains = NULL,
  drop = TRUE,
  ...
)
```

*Arguments:*

pars Optional character or numeric vector of parameter names or indices to extract. Supports special keywords: "parameters", "transform", "generate", and "all".  
 type Character string specifying the estimation type.  
 component Character string specifying the component to filter by.  
 chains Numeric vector of chains to include. If NULL, the best chain is used.  
 best\_chains Number of best chains to include.  
 drop Logical; if TRUE and only one parameter is selected, return the value directly instead of a list.  
 ... Additional arguments passed to draws().

*Returns:* A named list of point estimates, or a single value if 'drop = TRUE'.

**Method** EAP(): Calculate Expected A Posteriori (EAP) estimates from the best variational estimate by default.

*Usage:*

```
VB_Fit$EAP(
  pars = "parameters",
  chains = NULL,
  best_chains = NULL,
  drop = TRUE,
  ...
)
```

*Arguments:*

pars Optional character vector of parameter names to extract.  
 chains Numeric vector of chains to include. If NULL, the best chain is used.  
 best\_chains Number of best chains to include.  
 drop Logical; whether to drop the list if only one parameter is selected.  
 ... Additional arguments passed to 'estimate()'.

*Returns:* A named list of EAP estimates.

**Method** MAP(): Calculate Maximum A Posteriori (MAP) estimates from the best variational estimate by default.

*Usage:*

```
VB_Fit$MAP(  
  pars = "parameters",  
  chains = NULL,  
  best_chains = NULL,  
  type = c("marginal", "joint"),  
  drop = TRUE,  
  ...  
)
```

*Arguments:*

pars Optional character vector of parameter names to extract.

chains Numeric vector of chains to include. If NULL, the best chain is used.

best\_chains Number of best chains to include.

type Character string; "marginal" or "joint" MAP.

drop Logical; whether to drop the list if only one parameter is selected.

... Additional arguments passed to 'estimate()'.

*Returns:* A named list of MAP estimates.

**Method** new(): Create a new 'VB\_Fit' object.

*Usage:*

```
VB_Fit$new(  
  model,  
  fit,  
  random_fit,  
  elbo_history,  
  laplace,  
  posterior_mean,  
  ELBO,  
  rel_obj_vals,  
  best_chain,  
  mu_history  
)
```

*Arguments:*

model An 'RTMB\_Model' object.

fit A 3D array of parameter draws.

random\_fit A 3D array of random effect draws.

elbo\_history A list of numeric vectors of ELBO values for each chain.

laplace Logical; indicates if Laplace approximation was used.

posterior\_mean A named numeric vector of posterior means.

ELBO A numeric vector of final ELBO values for each chain.

rel\_obj\_vals A numeric vector of final relative objective tolerance values for each chain.

best\_chain Integer; the index of the chain with the maximum ELBO.

mu\_history Matrix of the parameter trajectory from the final window.

**Method** print(): Print a brief summary of the fitted object.

*Usage:*

```
VB_Fit$print(...)
```

*Arguments:*

... Additional arguments passed to the ‘summary’ method.

*Returns:* The object itself, invisibly.

**Method** draws(): Extract posterior draws for selected parameters.

*Usage:*

```
VB_Fit$draws(
  pars = NULL,
  chains = NULL,
  best_chains = NULL,
  inc_random = FALSE,
  inc_transform = TRUE,
  inc_generate = TRUE,
  best_only = FALSE
)
```

*Arguments:*

pars Character or numeric vector specifying the names or indices of parameters to extract. If NULL, all available parameters are extracted.

chains Numeric vector specifying the chains to extract. If NULL, draws from all chains are returned.

best\_chains Integer; number of best chains to retain based on ELBO. If supplied, chains with the highest ELBO are retained.

inc\_random Logical; whether to include random effects in the output. Default is FALSE.

inc\_transform Logical; whether to include transformed parameters in the output. Default is TRUE.

inc\_generate Logical; whether to include generated quantities in the output. Default is TRUE.

best\_only Logical; whether to extract only from the chain with the maximum ELBO. Default is FALSE unless explicitly requested.

*Returns:* A 3D array of posterior draws ‘[iterations, chains, parameters]’.

**Method** summary(): Summarize posterior draws.

*Usage:*

```
VB_Fit$summary(
  pars = NULL,
  max_rows = 10,
  digits = 2,
  inc_random = FALSE,
  inc_transform = TRUE,
  inc_generate = TRUE
)
```

*Arguments:*

**pars** Character or numeric vector specifying the names or indices of parameters to summarize.

If NULL, all available parameters are summarized.

**max\_rows** Integer; maximum number of rows to print in the summary table. Default is 10.

**digits** Integer; number of decimal places to print. Default is 2.

**inc\_random** Logical; whether to include random effects in the summary. Default is FALSE.

**inc\_transform** Logical; whether to include transformed parameters in the summary. Default is TRUE.

**inc\_generate** Logical; whether to include generated quantities in the summary. Default is TRUE.

*Returns:* A data frame containing the summarized posterior statistics.

**Method plot\_elbo():** Plot the ELBO history to diagnose convergence.

*Usage:*

```
VB_Fit$plot_elbo(tail_n = 1000, ests = NULL, type = "l", ...)
```

*Arguments:*

**tail\_n** Integer; the number of recent iterations to plot. If NULL, plots the entire history. Default is 2000.

**ests** Character string "best", numeric vector of estimate indices (e.g., 'c(1, 3)'), or 'NULL' to plot all. Default is 'NULL'.

**type** Character string; the type of plot. Default is "l" (lines).

... Additional arguments passed to the 'plot' function.

*Returns:* The object itself, invisibly.

**Method plot\_trajectory():** Plot the parameter trajectory from the final optimization window.

*Usage:*

```
VB_Fit$plot_trajectory(pars = NULL, type = "l", ...)
```

*Arguments:*

**pars** Character vector specifying the names of parameters to plot. If NULL, plots all available parameters.

**type** Character string; the type of plot. Default is "l" (lines).

... Additional arguments passed to the 'matplot' function.

*Returns:* The object itself, invisibly.

**Method transformed\_draws():** Compute transformed parameters from posterior draws.

*Usage:*

```
VB_Fit$transformed_draws(
  tran_fn = NULL,
  progress = c("auto", "none", "bar", "message"),
  tape = c("auto", "none", "force")
)
```

*Arguments:*

`tran_fn` An optional user-supplied function that takes data and parameter lists to return transformed quantities.

`progress` Progress reporting style: "auto", "none", "bar", or "message". "auto" and "bar" use line-based messages.

`tape` Derived-quantity evaluation mode. "auto" tries RTMB tape evaluation and falls back to R evaluation on error; "none" always uses R evaluation; "force" requires tape evaluation and errors if it fails.

*Returns:* The 'VB\_Fit' object itself, invisibly.

**Method** `generated_quantities()`: Compute generated quantities from posterior draws.

*Usage:*

```
VB_Fit$generated_quantities(
  code,
  progress = c("auto", "none", "bar", "message"),
  tape = c("auto", "none", "force")
)
```

*Arguments:*

`code` An 'rtmb\_code({ ... })' or '{ ... }' block containing the logic to be calculated using posterior samples.

`progress` Progress reporting style: "auto", "none", "bar", or "message". "auto" and "bar" use line-based messages.

`tape` Derived-quantity evaluation mode. "auto" tries RTMB tape evaluation and falls back to R evaluation on error; "none" always uses R evaluation; "force" requires tape evaluation and errors if it fails.

*Returns:* The 'VB\_Fit' object itself (invisibly). Results are appended to the 'generate\_fit' field.

**Method** `WAIC()`: Compute WAIC from pointwise generated log likelihood.

*Usage:*

```
VB_Fit$WAIC(...)
```

*Arguments:*

... Additional arguments passed to 'draws()', such as 'chains' or 'best\_chains'.

*Returns:* A 'waic\_BayesRTMB' object.

**Method** `diagnose()`: Run basic diagnostics for the variational fit.

*Usage:*

```
VB_Fit$diagnose(...)
```

*Arguments:*

... Additional arguments passed to 'diagnose\_vb\_fit()'.

*Returns:* A 'diagnose\_BayesRTMB' object.

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
VB_Fit$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

# Index

- \* **datasets**
  - beverage, [6](#)
  - BigFive, [7](#)
  - debate, [19](#)
  - training, [134](#)
- \* **distributions**
  - distributions, [21](#)
- \* **utilities**
  - math\_functions, [43](#)
- \* **wrappers**
  - rtmb\_wrappers, [123](#)
- ADVI\_method, [4](#)
- bayes\_factor, [5](#)
- BayesRTMB::RTMB\_Fit\_Base, [9](#), [38](#), [45](#), [136](#)
- beverage, [6](#)
- BigFive, [7](#)
- center\_grand\_mean, [8](#)
- center\_within\_cluster, [9](#)
- Classic\_Fit, [9](#)
- conditional\_effects, [17](#)
- conditional\_effects.mcmc\_fit, [18](#)
- debate, [19](#)
- diffusion\_lpdf, [19](#)
- Dim, [20](#)
- distance, [21](#)
- distributions, [21](#), [54](#), [75](#), [76](#), [119](#)
- ess\_basic, [23](#)
- ess\_bulk, [23](#)
- ess\_tail95, [24](#)
- exp\_mod\_normal\_lpdf, [24](#)
- fabs, [25](#), [119](#)
- gaussian\_process\_lpdf, [25](#)
- generate\_random\_init, [26](#)
- hist, [73](#)
- inv\_logit, [27](#), [75](#), [119](#)
- item\_curve, [27](#), [58](#)
- item\_curve.RTMB\_Fit\_Base, [28](#)
- item\_info, [28](#), [59](#)
- item\_info.RTMB\_Fit\_Base, [29](#)
- log1m, [29](#)
- log1m\_exp, [30](#)
- log1p\_exp, [30](#), [119](#)
- log\_det\_chol, [31](#), [119](#)
- log\_mix, [32](#), [119](#)
- log\_softmax, [32](#)
- log\_sum\_exp, [33](#), [75](#), [119](#)
- log\_sum\_exp\_matrix, [33](#)
- logit, [31](#)
- lsmeans, [34](#), [59](#)
- make\_bw\_from\_ydif, [34](#)
- make\_glm\_re\_terms, [35](#)
- make\_glm\_Z\_matrix, [36](#)
- make\_init\_mdu, [36](#)
- make\_ydif\_from\_bw, [37](#)
- map\_est, [38](#)
- MAP\_Fit, [38](#)
- math\_functions, [43](#), [54](#), [75](#), [76](#), [119](#)
- MCMC\_Fit, [45](#), [129](#)
- parameter\_types, [54](#), [75](#), [76](#)
- parameters\_code, [53](#)
- plot.ce\_rtmb, [55](#)
- plot.rtmb\_lsmeans, [55](#)
- plot.rtmb\_pp\_check (pp\_check), [63](#)
- plot\_acf, [56](#)
- plot\_conditional\_effects, [56](#)
- plot\_dens, [57](#)
- plot\_forest, [58](#)
- plot\_item\_curve, [58](#)
- plot\_item\_info, [59](#)

plot\_lsmeans, 59  
 plot\_mdu, 60  
 plot\_pairs, 61  
 plot\_test\_info, 62  
 plot\_trace, 62  
 posterior\_predict, 63  
 pp\_check, 63  
 print.bayes\_factor, 65  
 print.bayes\_factor\_rtmb, 65  
 print.ce\_rtmb, 66  
 print.ce\_simple, 66  
 print.summary\_BayesRTMB, 67  
 prior\_flat, 67  
 prior\_jzs, 68  
 prior\_normal, 68  
 prior\_rhs, 69  
 prior\_ssp, 70  
 prior\_uniform, 70  
 prior\_weak, 71  
  
 quad\_form\_chol, 71, 119  
 quad\_form\_diag, 72  
 quantile95, 72  
  
 r\_hat, 124  
 read\_mcmc\_csv, 73  
 restore\_bw\_from\_ydif  
     (make\_bw\_from\_ydif), 34  
 rhat\_summary, 73  
 rtmb\_array, 74  
 rtmb\_code, 54, 75, 119  
 rtmb\_corr, 77  
 rtmb\_fa, 78  
 RTMB\_Fit\_Base, 79  
 rtmb\_glm, 84  
 rtmb\_glmer, 87  
 rtmb\_irt, 90  
 rtmb\_lm, 92  
 rtmb\_lmer, 94  
 rtmb\_loglinear, 97  
 rtmb\_lrt, 98  
 rtmb\_mdu, 100  
 rtmb\_mediation, 102  
 rtmb\_mixture, 104  
 RTMB\_Model (RTMB\_Model-class), 108  
 rtmb\_model, 76, 105  
 RTMB\_Model-class, 108  
 rtmb\_syntax, 106, 119  
 rtmb\_table, 120  
  
 rtmb\_ttest, 121  
 rtmb\_vector, 122  
 rtmb\_wrappers, 123  
  
 safe\_rtmb\_model, 125  
 simple\_effects, 125  
 simple\_effects.mcmc\_fit, 126  
 softmax, 127  
 sort\_loadings, 127  
 squared\_distance, 128  
 stz\_basis, 128  
 sum\_to\_zero, 130  
 summary.ce\_rtmb, 129  
 summary\_mcmc, 129  
  
 test\_info, 62, 130  
 to\_centered\_matrix, 131  
 to\_centered\_tri, 131  
 to\_long, 132  
 to\_lower\_tri, 133  
 to\_wide, 133  
 training, 134  
 transform\_code, 134  
  
 upgrade\_fit, 135  
  
 validate\_data, 136  
 VB\_Fit, 136