Package: bml (via r-universe)

November 2, 2026

Type Package

Title Bayesian Multiple-Membership Multilevel Models with Parameterizable Weight Functions

Version 0.12.0

Description Implements Bayesian multiple-membership multilevel models with parameterizable weight functions via 'JAGS' to model how lower-level units jointly shape higher-level outcomes (micro-macro link) across a range of outcome types (e.g., linear, logit, and survival models). Supports estimation and comparison of alternative aggregation mechanisms, allows weight matrices to be endogenized through parameters and covariates, and accommodates complex dependence structures that extend beyond traditional multilevel frameworks. For details, see Rosche (2026) ``A Multilevel Model for Coalition Governments. Uncovering Party-Level Dependencies Within and Between Governments" <doi:10.31235/osf.io/4bafr_v2>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports dplyr, generics, readr, tidyr, stringr, R2jags, ggplot2, ggmcmc, coda, patchwork, tibble, purrr, rlang, posterior, loo, nlme, parallel, stats, utils

URL https://benrosche.github.io/bml/

BugReports https://github.com/benrosche/bml/issues

RoxygenNote 7.3.3

Suggests rmarkdown, knitr, testthat (>= 3.0.0), rjags, bayesplot

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs jags libicu-dev libssl-dev libx11-dev

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

Date/Publication 2026-08-04 11:50:31 UTC

RemoteUrl https://github.com/benrosche/bml

RemoteRef HEAD

RemoteSha b5ef5d571b235a258bc4191797b4ff9c40345c55

Contents

as_draws.bml . . . 3
bml . . . 3
bml-families . . . 7
bml_delta . . . 8
bml_realloc . . . 9
bml_turnover . . . 9
bmlCompare . . . 10
coalgov . . . 12
coef.bml . . . 15
coefPlot . . . 16
conditional_effects . . . 17
est . . . 18
fe . . . 19
fitted.bml . . . 19
fix . . . 20
fixef.bml . . . 21
fn . . . 21
get_prior . . . 22
glance.bml . . . 23
hm . . . 24
id . . . 26
log_lik . . . 27
loo.bml . . . 27
make_jagscode . . . 28
mcmcDiag . . . 29
mm . . . 31
monetPlot . . . 33
posterior_predict . . . 35
pp_check . . . 35
predict.bml . . . 36
prior . . . 37
ranef.bml . . . 38
re . . . 38
summary.bml . . . 39
tidy.bml . . . 41
varDecomp . . . 43
vars . . . 45
w . . . 46
waic.bml . . . 48

Index **49**

`as_draws.bml` *Extract posterior draws from a bml model*

Description

Converts the retained MCMC chains into the **posterior** package's draws formats, unlocking the whole **posterior/bayesplot** toolchain (`summarise_draws()`, `rhat()`, `ess_bulk()`, ...). Variables keep their internal JAGS node names (`b[1]`, `b.fn.1[1]`, `sigma.mm.1`, ...); the mapping to term labels is in `model$reg.table` (rownames = node, `Parameter` = label).

Usage

```
## S3 method for class 'bml'
as_draws(x, ...)

## S3 method for class 'bml'
as_draws_df(x, ...)

## S3 method for class 'bml'
as_draws_matrix(x, ...)

## S3 method for class 'bml'
as_draws_array(x, ...)
```

Arguments

`x`	A fitted `bml` model with retained posterior draws.
`...`	Passed on to the **posterior** converters.

Value

A `draws_array` (or the format of the called variant) containing the capabilities requested when the model was fitted.

`bml` *Bayesian Multilevel Models for Micro-Macro Analysis (Multiple Membership) Using JAGS*

Description

The **bml** package estimates micro-to-macro regressions: Bayesian multiple-membership multilevel models in which the aggregation from member-level records to group-level outcomes is an explicit, estimable object — weights (`w`), aggregation functions (`fn`: mean, variance, concentration, thresholds, smooth max/min, CES means, covariance, or a user-written reduction), member/unit random and fixed effects (`re`, `fe`), and cross-level or feature-by-feature interactions via named blocks.

JAGS must be installed separately: https://sourceforge.net/projects/mcmc-jags/.

Usage

```
bml(
  formula,
  data,
  family = stats::gaussian(),
  prior = NULL,
  inits = NULL,
  iter = 1000,
  warmup = 500,
  thin = max(1, floor((iter - warmup)/1000)),
  chains = 3,
  cores = 1,
  n.adapt = 1000,
  seed = NULL,
  run = TRUE,
  monitor = "summary",
  modelfile = FALSE,
  ...
)
```

Arguments

formula	A symbolic model formula; see Details.
data	Data frame in member-level (long) format: one row per membership. Must contain all variables referenced in the formula, including the identifiers in `id()`.
family	Model family: a family function (`gaussian()`, `bernoulli()`, `weibull()`, `cox()`) or a string (`"gaussian"`, `"bernoulli"`, `"weibull"`, `"cox"`). Cox baseline-hazard intervals travel with the family: `cox(intervals = 10)`.
prior	Prior specifications built with `prior` and combined with `c()`; see `get_prior` for the settable parameters. Raw JAGS strings (e.g. `"b.fn.1[1] ~ dnorm(0, 0.1)"`) pass through untranslated as an escape hatch.
inits	List of initial values for MCMC chains (applied to all chains). Weight and `fn` shape parameters are started from dispersed values drawn separately for each chain, as R-hat assumes; user values override and are applied as given.
iter	Total number of MCMC iterations per chain. Default: 1000.
warmup	Number of warmup (burn-in) iterations discarded from the start of each chain. Default: 500.
thin	Thinning rate. Default targets ~1000 retained draws.
chains	Number of MCMC chains. Default: 3.
cores	Number of cores; `cores > 1` runs chains in parallel. Default: 1.
n.adapt	Number of JAGS adaptation iterations used to tune the samplers before warmup. Tuning is frozen once adaptation ends, so models with non-conjugate nodes (estimated `w()` or `fn()` parameters) benefit from a longer adaptation phase. Default: 1000.
seed	Integer random seed for reproducibility.

run	Logical; if FALSE, returns the generated JAGS model string, data, and monitors without fitting (see also make_jagscode).
monitor	Character vector selecting retained posterior capabilities. The default, "summary", retains posterior summaries and convergence diagnostics but no draws. Add "parameters", "random_effects", "weights", "fitted", "predictive", or "log_lik" as needed. "full" stores all family-supported draws once in a compact posterior::draws_array; "jags" explicitly retains the raw R2jags object.
modelfile	Logical or character path: TRUE saves the generated JAGS code to modelstring.txt; a path reads JAGS code from that file instead of generating it.
...	Not used; catches removed legacy arguments (n.iter, n.burnin, n.thin, n.chains, parallel, priors, cox_intervals) with a migration message.

Details

The general formula structure is

```
outcome ~ 1 + predictors + mm(...) + hm(...)
```

where mm defines a multiple-membership block (the micro-macro link) and hm a hierarchical nesting level. Multiple mm() blocks stack features (e.g. mean + variance + concentration); multiple hm() blocks give cross-classification. For survival families use Surv(time, event) on the left-hand side.

Named mm() blocks can be referenced in main-formula interactions:

```
Y ~ education + Ax:education +
 mm(name = Ax, id = id(task, occ), vars = vars(x), w = w(~ importance), fn = fn("sum"))
```

Both cross-level (feature x macro variable) and block x block (feature x feature) interactions are supported; the macro variable must also appear as a main effect, and a bare (non-interacted) block reference is an error.

The package is introduced in Rosche (2026), *Political Analysis*.

Value

A list of class "bml" with elements reg.table (posterior summaries; labeled terms), w (weight matrices per block), re.mm/re.hm (random effects), pred (posterior predictive means), fitted (posterior means of the linear predictor), input (model metadata), diagnostics, storage, draws (requested compact posterior draws), and jags.out (non-NULL only when monitor = "jags").

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

References

Rosche, B. (2026). A Multilevel Model for Coalition Governments: Uncovering Party-Level Dependencies Within and Between Governments. *Political Analysis*.

Browne, W. J., Goldstein, H., & Rasbash, J. (2001). Multiple membership multiple classification (MMMC) models. *Statistical Modelling*, 1(2), 103-124.

See Also

mm, hm, w, fn, re, fe, prior, get_prior, summary.bml, fixef.bml, ranef.bml, loo.bml, varDecomp, bmlCompare, make_jagscode

Examples

```
## Not run:
data(coalgov)

# Additive aggregation with member random effects, country nesting
m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE) +
     hm(id = id(cid)),
  data = coalgov,
  family = weibull()
)
summary(m1)

# Estimated weights (b0, b1 are free parameters by the one-parameter rule)
m2 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion),
       w = w(~ ilogit(b0 + b1 * pseat), scale = TRUE),
       fn = fn("sum")),
  data = coalgov,
  family = weibull()
)

# Emergent features: variance + concentration, stacked
m3 <- bml(
  sim.y ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("var")) +
    mm(id = id(pid, gid), w = w(~ pseat, scale = TRUE), fn = fn("hhi")),
  data = coalgov,
  family = gaussian()
)

# Structured priors
m4 <- bml(
  sim.y ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE),
  data = coalgov,
  family = gaussian(),
  prior = c(
    prior(normal(0, 5), class = "b"),
    prior(exponential(1), class = "sd")
  )
)

## End(Not run)
```

bml-families *Model families for bml*

Description

Family functions specify the outcome distribution and link function of a bml model, following the **brms**/glm convention of passing a family *function* rather than a string:

- gaussian(): normal outcome, identity link
- bernoulli() (alias binomial()): binary outcome, logit link
- weibull(): Weibull survival model; outcome Surv(time, event)
- cox(intervals = NULL): Cox proportional hazards model; outcome Surv(time, event). intervals controls the baseline hazard: NULL uses all unique event times (non-parametric); an integer k uses a piecewise-constant baseline hazard with k intervals (faster).

Family *strings* are accepted as first-class input too (as in brms): family = "gaussian", "bernoulli", "weibull", "cox".

Usage

```
bernoulli(link = "logit")

weibull(link = "log")

cox(intervals = NULL, link = "log")
```

Arguments

link	Link function. Only the canonical links are supported by the JAGS backend ("identity" for gaussian, "logit" for bernoulli, "log" for weibull/cox); passing anything else errors.
intervals	For cox(): NULL for a non-parametric baseline hazard (all unique event times), or a positive integer number of intervals for a piecewise-constant baseline hazard.

Value

A bml_family object (a list with elements family and link, plus intervals for cox()).

Examples

```
## Not run:
bml(y ~ x + mm(...), data = dat, family = gaussian())
bml(y01 ~ x + mm(...), data = dat, family = bernoulli())
bml(Surv(t, ev) ~ x + mm(...), data = dat, family = weibull())
bml(Surv(t, ev) ~ x + mm(...), data = dat, family = cox(intervals = 10))

## End(Not run)
```

`bml_delta` *Temporal composition change: matched decomposition of a feature difference*

Description

Computes the change in a group's weighted mean attribute between two snapshots, $\Delta A_x = A_{x,t_2} - A_{x,t_1}$, together with its matched (shift-share / Foster-Haltiwanger-Krizan style) decomposition over membership spells:

$$\Delta A_x = \underbrace{\sum_k \Delta w_k \bar{x}_k}_{reallocation} + \underbrace{\sum_k \bar{w}_k \Delta x_k}_{attribute\ change} \quad + entry/exit\ terms$$

The components sum to ΔA_x exactly (adding-up identity). The returned group-level covariates can be entered into a `bml` main formula.

Usage

```
bml_delta(data, group, member, time, weight = NULL, x, t1, t2)
```

Arguments

`data`	Member-level long data covering both snapshots.
`group`, `member`, `time`	Column names (strings): group id, member id, and the time variable.
`weight`	Column name (string) of the raw weight, or `NULL` for equal weights.
`x`	Column name (string) of the member attribute.
`t1`, `t2`	The two time values to compare.

Details

Weights are normalized within each group and snapshot, so Δw sums to zero over the union roster. For stayers, $\bar{x}$ and $\bar{w}$ are the across-snapshot means; entering/exiting members contribute their full wx term to `entry_exit`.

Value

A data frame with one row per group: `dA` (total change), `realloc` (reallocation among stayers), `attr_change` (attribute change among stayers), `entry_exit` (net entry/exit contribution).

See Also

`bml_realloc`, `bml_turnover`

bml_realloc *Reallocation of membership weights toward high-attribute members*

Description

The headline temporal micro-macro term: did the group shift weight mass toward members with high x? Computes $\sum_k \Delta w_k x_k$ over the union roster (using each member's observed attribute) plus the reallocation *intensity* $\sum_k \bar{w}_k (\Delta w_k)^2$-style spread measures.

Usage

```
bml_realloc(data, group, member, time, weight = NULL, x, t1, t2)
```

Arguments

`data`	Member-level long data covering both snapshots.
`group`, `member`, `time`	Column names (strings): group id, member id, and the time variable.
`weight`	Column name (string) of the raw weight, or `NULL` for equal weights.
`x`	Column name (string) of the member attribute.
`t1`, `t2`	The two time values to compare.

Value

A data frame with one row per group: `realloc_x` ($\sum_k \Delta w_k x_k$; reallocation toward high-`x` members) and `realloc_intensity` ($\sum_k (\Delta w_k)^2$; how much reshuffling happened, regardless of direction).

See Also

`bml_delta`, `bml_turnover`

bml_turnover *Membership turnover between two snapshots*

Description

Set-level composition dynamics: how much of the group's membership mass entered or exited between the two snapshots.

Usage

```
bml_turnover(data, group, member, time, weight = NULL, t1, t2, x = NULL)
```

Arguments

data	Member-level long data covering both snapshots.
group, member, time	Column names (strings): group id, member id, and the time variable.
weight	Column name (string) of the raw weight, or NULL for equal weights.
t1, t2	The two time values to compare.
x	Not used (turnover is a property of the roster and weights alone).

Value

A data frame with one row per group: entry_mass (weight mass of entrants at t2), exit_mass (weight mass of exiters at t1), jaccard_dist (1 - |intersection| / |union| of the member sets), and realloc_l1 ($\frac{1}{2}\sum_k |\Delta w_k|$, the total variation distance between the two weight measures).

See Also

bml_delta, bml_realloc

---

bmlCompare *Compare fitted bml models in a publication-style table*

Description

Builds a side-by-side regression table from one or more fitted bml models: one row per term, one column per model, with cells formatted as estimate (conf.low, conf.high). Goodness-of-fit rows (e.g. N and DIC) are appended at the bottom. The result is a data.frame of character cells, ready to pass to knitr::kable(), gt::gt(), etc.

For the underlying numeric values (e.g. to build custom tables or plots) use the **broom** methods tidy.bml and glance.bml directly; they also make modelsummary::modelsummary(list(m1, m2)) work on bml models.

Usage

```
bmlCompare(
  ...,
  component = "all",
  terms = NULL,
  labels = NULL,
  stats = c("N", "DIC", "Convergence"),
  digits = 3,
  conf.level = 0.95
)
```

Arguments

...	Fitted bml model objects, optionally named (e.g., bmlCompare(base = m1, weighted = m2)). The names become the column headers. Unnamed models are labeled with the expressions they were passed as (e.g., "m1").
component	Which parameters to include; passed to tidy.bml. One of "all" (default), "fixed", "random", or "weights".
terms	Optional character vector of term names selecting which rows to show and in what order (matched against the term labels from tidy.bml, e.g. "smoking_alter (mm.1)"). Terms not found are skipped with a warning. Default NULL: all terms in component, in their natural order.
labels	Optional character vector renaming the rows for display. Must be the same length as the selected terms (so supply terms when using labels). Default NULL: keep the original term labels.
stats	Goodness-of-fit rows to append, in order. Any of "N" (main-level units), "n.members" (member-level units), "DIC", and "Convergence". Default: c("N", "DIC", "Convergence"). Use character(0) for none.
digits	Number of decimal places for the estimate and interval bounds. Default: 3.
conf.level	Width of the equal-tailed credible interval. Default: 0.95. Other levels require the models to use monitor = "parameters" (see tidy.bml).

Value

A data.frame whose first column Term holds the term labels (and the requested stats labels), followed by one character column per model containing estimate (conf.low, conf.high) cells.

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

bml, tidy.bml, glance.bml, coefPlot

Examples

```
## Not run:
data(coalgov)

m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE),
  family = weibull(),
  data = coalgov
)
m2 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
   mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ pseat), fn = fn("sum"), RE = TRUE),
  family = weibull(),
```

```
  data = coalgov
)

bmlCompare("Equal" = m1, "Seat share" = m2)

# Select and relabel specific rows
bmlCompare(
  "Equal" = m1, "Seat share" = m2,
  terms  = c("majority", "cohesion (mm.1)"),
  labels = c("Majority", "Cohesion")
) |> knitr::kable()

## End(Not run)
```

`coalgov` *Coalition Governments in Western Democracies (1944-2014)*

Description

A dataset containing information on coalition governments and their member parties across 30 parliamentary democracies. The data are in long format where the unit of analysis is parties in governments, making it suitable for multiple-membership multilevel models where governments (groups) are composed of multiple parties (members).

Usage

```
coalgov
```

Format

A tibble with 2,077 rows and 18 variables. Each row represents a party's participation in a specific coalition government. The sample contains 628 governments formed by 312 unique parties across 29 countries.

Identifiers:

gid Government identifier (group-level unit in `mm()` specification). Range: [3, 1105]

pid Party identifier (member-level unit in `mm()` specification). Range: [11110, 96955]

cid Country identifier (nesting-level unit in `hm()` specification). Range: [11, 96]

cname Three-letter country code (ISO 3166-1 alpha-3)

pname Full party name

Government-level variables:

election Date of the preceding election that led to the government's formation. Range: [1939-04-02, 2014-12-14]

n Number of parties in the coalition (group size for weight functions). Range: [2, 9], mean: 3.31

dur_wkb Government duration in days, measured from investiture to termination (outcome variable for survival models). Range: [7, 1840], mean: 554.5

event_wkb Early termination indicator: 1 = government terminated due to political conflict (voluntary resignation, dissension within government, lack of parliamentary support, or head of state intervention) more than one year before the official end of term; 0 = censored (regular elections, other reasons, or termination within one year of scheduled elections). Range: [0, 1], mean: 0.39

majority Majority government indicator: 1 = coalition controls majority of parliamentary seats, 0 = minority government. Range: [0, 1], mean: 0.80

mwc Minimal winning coalition indicator: 1 = coalition would lose its majority if any party left, 0 = oversized coalition. Range: [0, 1], mean: 0.35

rile_SD Inter-party ideological heterogeneity. Standard deviation of coalition parties' left-right positions (from CMP) relative to the ideological distribution of all parties in parliament. Standardized and inverted so higher values indicate greater ideological cohesion. Range: [-8.40, 2.12], mean: 0.04

Country-level variables:

investiture Investiture vote requirement (time-constant country characteristic): 1 = country requires formal parliamentary investiture vote, 0 = no formal requirement. Range: [0, 1], mean: 0.46

Party-level variables:

pseat Party's relative seat share within the coalition, computed as `pseat / sum(pseat)` within each government. Sums to 1 within each coalition. Range: [0.00, 1.00], mean: 0.33

prime Prime minister party indicator: `TRUE` = party holds prime ministership (n = 628), `FALSE` = junior coalition partner (n = 1,449)

cohesion Intra-party ideological cohesion, measured using an adaptation of the Cowles-Jones ratio. Computed as the ratio of continuous ideological shifts to reversals in a party's left-right position over time. Higher values indicate more consistent ideological trajectories (greater cohesion). Standardized. Range: [-1.13, 3.85], mean: 0.00

rile Party's left-right ideological position (from CMP). Measured on a continuous scale where higher values indicate more right-wing positions and lower values indicate more left-wing positions. Standardized. Range: [-3.21, 3.68], mean: 0.00

finance Party's economic dependence on member contributions (from PPDB). Measured as the share of party funding from member dues relative to total income. Standardized; higher values indicate greater dependence on member financing. Treated as time-constant due to data limitations. Range: [-0.98, 4.40], mean: 0.00

Nmembers Number of party members (from PPDB). Standardized; treated as time-constant due to data limitations. Range: [-0.33, 15.02], mean: 0.00

Details

This dataset demonstrates multiple-membership multilevel modeling where:

- **Members:** Political parties (identified by `pid`)
- **Groups:** Coalition governments (identified by `gid`)

- **Nesting:** Governments nested within countries (identified by `cid`)

Each coalition government comprises multiple parties, and parties can participate in multiple governments over time. This creates a multiple-membership structure where party-level characteristics are aggregated to the government level using weighting functions specified in `mm()` blocks.

Sample: After matching party data across sources and excluding single-party and caretaker governments, the sample comprises 628 governments formed by 312 unique parties across 29 countries: Australia, Austria, Belgium, Bulgaria, Croatia, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Israel, Italy, Japan, Latvia, Lithuania, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Spain, Sweden, Switzerland, and United Kingdom.

Measurement notes:

- Government duration follows the WKB convention: time from investiture to termination or new elections
- Early termination events focus on political gridlock (conflict-related endings) and exclude terminations within one year of scheduled elections
- Party-level variables (`cohesion`, `finance`, `Nmembers`) are standardized (mean = 0) for analysis

Source

Data compiled from multiple sources:

- **Coalition governments:** Woldendorp, Keman, and Budge (WKB) dataset, updated by Seki and Williams (2014)
- **Party ideology:** Comparative Manifesto Project (CMP; Volkens et al. 2016)
- **Party organization:** Political Party Database (PPDB; Scarrow, Poguntke, and Webb 2017)

Missing party-level data imputed using multiple imputation by chained equations with predictive mean matching.

References

Seki, K., & Williams, L. K. (2014). Updating the Party Government data set. *Electoral Studies*, 34, 270-279.

Volkens, A., et al. (2016). The Manifesto Data Collection. Manifesto Project (MRG/CMP/MARPOR). Version 2016a. Berlin: Wissenschaftszentrum Berlin fur Sozialforschung.

Scarrow, S. E., Webb, P. D., & Poguntke, T. (Eds.). (2017). *Organizing Political Parties: Representation, Participation, and Power*. Oxford University Press.

See Also

`bml` for modeling examples using this dataset

Examples

```
data(coalgov)

# Explore data structure
str(coalgov)
table(coalgov$cname)

# Number of unique units
length(unique(coalgov$gid))   # Governments
length(unique(coalgov$pid))   # Parties
length(unique(coalgov$cid))   # Countries

## Not run:
# Model: government duration as function of majority status and party characteristics
m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(finance), w = w(~ 1/n), fn = fn("sum"), RE = TRUE) +
    hm(id = id(cid)),
  family = weibull(),
  data = coalgov
)
summary(m1)

## End(Not run)
```

`coef.bml` *Extract combined coefficients from a bml model*

Description

Following brms semantics loosely: the fixed coefficients plus, per grouping structure, the unit-level effects (fixed part + random deviation).

Usage

```
## S3 method for class 'bml'
coef(object, ...)
```

Arguments

`object`	A fitted `bml` model.
`...`	Unused.

Value

A list with elements `fixef` (matrix) and `ranef` (list).

coefPlot *Coefficient plot for fitted bml models*

Description

Creates a coefficient ("dot-and-whisker") plot from one or more fitted `bml` models: posterior means as points and credible intervals as horizontal ranges, with a dashed reference line at zero. When several models are supplied, their coefficients are placed next to each other for each term so the models can be compared directly.

Usage

```
coefPlot(
  ...,
  parameters = NULL,
  intercept = TRUE,
  conf.level = 0.95,
  component = "fixed"
)
```

Arguments

`...`	One or more fitted `bml` model objects, optionally named (e.g., `coefPlot(base = m1, weighted = m2)`). Names are used in the legend; unnamed models are labeled with the expressions they were passed as (e.g., `"m1"`).
`parameters`	Optional character vector of term labels (as shown in `summary()`) to include, in the order they should appear (top to bottom). Default: all terms of the selected `component`.
`intercept`	Logical. Include the intercept? Default: `TRUE`.
`conf.level`	Width of the equal-tailed credible interval. Default: 0.95. Other levels require the models to use `monitor = "parameters"` (see `tidy.bml`).
`component`	Which parameters to plot; passed to `tidy.bml`. Default: `"fixed"` (regression coefficients).

Details

The plot uses a colorblind-friendly palette (Okabe-Ito) to distinguish models and intentionally substantial line widths and point sizes for readability in presentations and publications. All defaults can be overridden by adding **ggplot2** layers to the returned object, e.g. `coefPlot(m1) + ggplot2::scale_colour_brewer(palette = "Dark2")`.

Value

A `ggplot` object that can be further customized with **ggplot2** layers or saved with `ggplot2::ggsave()`.

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

bml, tidy.bml, bmlCompare, monetPlot

Examples

```
## Not run:
data(coalgov)

m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE),
  family = weibull(),
  data = coalgov
)

# Single model
coefPlot(m1)

# Without intercept
coefPlot(m1, intercept = FALSE)

# Compare two models side by side
coefPlot(equal = m1, seatshare = m2)

# Save the plot
p <- coefPlot(m1, m2)
ggplot2::ggsave("coefplot.pdf", p, width = 6, height = 4)

## End(Not run)
```

---

conditional_effects *Conditional effects of main-formula predictors*

Description

Marginal-effect curves for main-formula covariates: the expected outcome on the linear-predictor scale across a grid of the chosen predictor, holding the other main-formula covariates at their means and the block features, random effects, and interactions at their sample-average contribution.

Usage

```
conditional_effects(x, ...)

## S3 method for class 'bml'
conditional_effects(x, effects = NULL, resolution = 100, ...)
```

Arguments

x	A fitted bml model with monitor = c("parameters", "fitted").
...	Unused.
effects	Character vector of main-formula term names. Default: all numeric main-formula terms.
resolution	Grid size. Default: 100.

Value

A named list of data frames (one per effect) with columns value, estimate, lb, ub, suitable for plotting.

---

est *Mark a shape parameter as estimated*

Description

Used inside fn's named-shortcut arguments to say "estimate this parameter" rather than fixing it to a number: fn("threshold", c = est(), kappa = 10). In the formula DSL a free symbol alone declares a parameter, so est() is only needed where a value slot must distinguish estimate-vs-fix.

Usage

```
est(lower = NULL, upper = NULL)
```

Arguments

lower, upper	Optional bounds; if both given, the default prior is dunif(lower, upper). Otherwise the parameter gets a parameter-specific default (e.g. the observed range of x for a threshold cutpoint). Override via prior(..., class = "fn").

Value

A bml_est marker object.

See Also

fn, prior

fe *Fixed effects: no pooling*

Description

The no-pooling peer of `re`: identical formula grammar, but unit effects get flat independent priors instead of a hierarchical prior (Gelman-Hill: "FE = RE with infinite variance"). `fe(1)` gives classic unit dummies; `fe(1 + x)` additionally gives unit-specific slopes. The first unit is the reference category (its effects are fixed to 0), so `fe()` remains identified alongside a global intercept and global slopes.

Usage

```
fe(x = 1, showFE = FALSE)
```

Arguments

`x`	The effects expression: `1`, `1 + var`, `0 + var`.
`showFE`	Logical; if `TRUE`, per-unit estimates are monitored and reported. Default `FALSE`.

Value

A `bml_fe` object.

See Also

`re`, `hm`, `mm`

fitted.bml *Posterior expectations of the linear predictor*

Description

Posterior expectations of the linear predictor

Usage

```
## S3 method for class 'bml'
fitted(object, summary = TRUE, ...)
```

Arguments

`object`	A fitted `bml` model with `monitor = "fitted"`.
`summary`	Logical; if `TRUE` (default), return a summary matrix (one row per observation), else the draws matrix (draws x observations).
`...`	Unused.

Value

A matrix of posterior summaries of mu, or a draws matrix.

`fix` *Fix a coefficient to a known value*

Description

Specify a covariate whose coefficient should be held constant at a fixed value rather than estimated from the data. This is useful for offset variables or when you want to impose theoretical constraints. Fixed coefficients are handled efficiently by pre-computing their contribution in R before passing data to JAGS.

Usage

```
fix(var, value)
```

Arguments

var	Unquoted variable name from your data
value	Numeric value for the coefficient (e.g., 1.0 for a standard offset)

Value

A bml_fix object that can be used within vars.

See Also

vars, mm

Examples

```
## Not run:
# Fix a coefficient to 1.0 (standard offset)
fix(exposure, 1.0)

# Use within vars() for multiple-membership models
vars(fix(population, 0.5) + income + education)

## End(Not run)
```

fixef.bml — *Extract fixed coefficients from a bml model*

Description

Returns all class-"b" coefficients — main-formula terms, block features, and named-feature interactions — labeled by term, one row per coefficient. fn shape parameters are deliberately excluded (they are not coefficients; see the "Function parameters" section of summary.bml).

Usage

```
## S3 method for class 'bml'
fixef(object, ...)
```

Arguments

object	A fitted bml model.
...	Unused.

Value

A matrix with columns Estimate, Est.Error, Q2.5, Q97.5 and term rownames.

fn — *The aggregation function of an mm() block*

Description

Selects the function f that reduces the weighted member records to a group-level feature — the f in the framework's $\theta^{micro,f}(M_{it})$. fn("sum") (the default) is the additive weighted mean; the other named types are "emergent" features of the whole member set. fn() also accepts a one-sided formula written in a restricted expression DSL, so users can define their own reductions.

Named types:

```
fn("sum")                          # A_x  = sum_k w_k x_k          (default)
fn("var")                          # V_x  = sum_k w_k (x_k - A_x)^2
fn("var", moment = 3)              # higher central moments
fn("hhi")                          # C    = sum_k w_k^2  (weights only)
fn("effn")                         # 1 / C
fn("entropy")                      # -sum_k w_k log w_k
fn("threshold", c = 0.7, kappa = 10)  # T(c) = sum_k w_k ilogit(kappa (x_k - c))
fn("threshold", c = est())         # estimate the cutpoint
fn("smax", kappa = 5)              # (1/kappa) log sum_k w_k exp(kappa x_k)
fn("smax", kappa = est())          # estimate the aggregation function itself:
                                   #   kappa<0 -> min, kappa->0 -> mean, kappa>0 -> max
fn("gmean", p = est())             # power/CES mean (sum_k w_k x_k^p)^(1/p); x > 0
fn("cov")                  # C_xz = sum_k w_k (x_k - A_x)(z_k - A_z); two attributes
```

Expression DSL: `w()` normalizes weights to a probability measure over the group's members; `E(e)` is the expectation $\sum_k w_k e_k$ under that measure. Member-level quantities are the `vars()` attributes and the reserved symbol `w`; anything wrapped in `E()` is a group scalar. Whitelisted operations: `+ - * / ^`, `exp`, `log`, `ilogit`, `pow`. Any symbol that is not a data column or reserved word is a *free parameter* with a default prior (one rule, shared with `w`); the build messages the detected parameters.

```
fn(~ E((x - E(x))^2))                    # variance, written out
fn(~ E(ilogit(kappa * (x - c))))         # threshold with free c, kappa
fn(~ E((x - E(x)) * (z - E(z))))         # covariance
```

Identification: internal parameters reach the outcome only through `b * feature`; if the feature's coefficient is ~0 they are unidentified. Dispersion functions need real within-group spread; threshold/tail functions need mass of `x` in the region they read. A free parameter that multiplies an attribute inside a nonlinear `E()` is confounded with the feature's coefficient — treat inner parameters as shape parameters (cutpoints, sharpness), not slopes.

Usage

```
fn(type = "sum", ...)
```

Arguments

`type`	A type string (one of `"sum"`, `"var"`, `"hhi"`, `"effn"`, `"entropy"`, `"threshold"`, `"smax"`, `"gmean"`, `"cov"`) or a one-sided formula (`~ E(...)`).
`...`	Shape parameters for the named types: `moment` (var), `c` and `kappa` (threshold), `kappa` (smax), `p` (gmean). Each is a number (fixed) or `est()` (estimated).

Value

A `bml_fn` object.

See Also

`mm`, `w`, `est`, `prior`

---

get_prior *List every settable prior of a bml model*

Description

Returns the table of all parameters whose priors can be set via `prior`, together with their default priors — the bml analogue of `brms::get_prior()`. Use it to discover valid `class`/`coef`/ `group` combinations before overriding.

Usage

```
get_prior(formula, data, family = stats::gaussian())
```

Arguments

`formula`	The model formula (same as for `bml`).
`data`	The data (member-level long format).
`family`	Model family (function, object, or string). Default `gaussian()`.

Value

A data frame with columns `class`, `coef`, `group`, `node` (the internal JAGS node), and `default`.

See Also

`prior`, `bml`

Examples

```
## Not run:
get_prior(
  y ~ x + mm(id = id(pid, gid), vars = vars(z), w = w(~ 1/n), fn = fn("sum")),
  data = dat, family = gaussian()
)

## End(Not run)
```

`glance.bml` *Glance at a fitted bml model*

Description

Returns a one-row tibble with model-level information following the **broom** conventions, suitable for model comparison.

Usage

```
## S3 method for class 'bml'
glance(x, ...)
```

Arguments

`x`	A fitted model object of class `"bml"` returned by `bml`.
`...`	Additional arguments (currently unused).

Value

A one-row `tibble` with columns:

- `nobs`: Number of main-level (outcome) units
- `n.members`: Number of member-level units across mm blocks
- `DIC`: Deviance Information Criterion
- `family`: Outcome family
- `iter`, `chains`: MCMC settings
- `storage`, `ndraws`: retained posterior information
- `max_rhat`, `min_ess_bulk`, `min_ess_tail`, `n_flagged`, and `convergence`: model-level diagnostics

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

`bml`, `tidy.bml`, `bmlCompare`

Examples

```
## Not run:
glance(m1)

## End(Not run)
```

`hm` *Define a hierarchical nesting structure*

Description

Specifies a hierarchical (nesting) level: each group-level unit belongs to exactly one nesting-level unit (e.g., governments within countries). `hm()` carries only the effect *structure* — random effects (`RE = re(...)`, partial pooling) or fixed effects (`FE = fe(...)`, no pooling). Fixed effects of unit-level *covariates* belong in the main formula, following the lme4/brms convention:

```
Y ~ gdp + hm(id = id(cid), RE = re(1 + gdp))   # like y ~ gdp + (1 + gdp | cid)
```

Usage

```
hm(id, RE = NULL, FE = NULL, labels = NULL, ...)
```

Arguments

id	An id object: id(hmid).
RE	Random effects: TRUE (shorthand for re(1)), an re object, or NULL. Slope variables are main-level columns varying within the nesting units. Default when neither RE nor FE is given: re(1).
FE	Fixed effects: a fe object (unit dummies via fe(1); unit-specific slopes via fe(1 + x); report per-unit estimates with fe(1, showFE = TRUE)). Mutually exclusive with RE.
labels	Unquoted variable name holding display labels for the nesting units (used when reporting per-unit fixed effects).
...	Not used; catches the removed type =, vars =, name =, showFE =, and ar = arguments (the last moved into the effects grammar: RE = re(1, ar = TRUE)) with a migration message.

Details

Cross-classified structures are modeled by including multiple hm() blocks. The old type = "RE"/"FE" and vars = arguments were removed: use RE = re(...) / FE = fe(...), and move unit-level covariates into the main formula (their coefficients are ordinary main-model coefficients).

Value

A bml_hm object.

References

Goldstein, H. (2011). *Multilevel Statistical Models* (4th ed.). Wiley.

See Also

bml, mm, id, re, fe

Examples

```
## Not run:
hm(id = id(cid))                          # random intercepts (default)
hm(id = id(cid), RE = re(1 + x))          # random intercept + slope on x
hm(id = id(cid), FE = fe(1))              # country dummies (no pooling)
hm(id = id(cid), FE = fe(1, showFE = TRUE), labels = cname)

## End(Not run)
```

`id` *Specify identifier variables for multiple-membership and hierarchical structures*

Description

Helper function used within `mm` and `hm` to specify the identifier variables that define memberships and nesting structures. In multiple-membership models, `id()` links member-level units (e.g., task IDs) to group-level units (e.g., occupation IDs) in long format: one row per membership, any number of members per group. In hierarchical models, `id()` specifies the nesting-level identifier (e.g., country ID).

Usage

```
id(...)
```

Arguments

`...`	Unquoted variable names from your data: • For `mm()`: Two identifiers `id(mmid, mainid)` where `mmid` identifies member-level units and `mainid` identifies group-level units • For `hm()`: One identifier `id(hmid)` where `hmid` identifies nesting-level units

Value

A `bml_id` object containing the variable names as character strings.

See Also

`mm`, `hm`, `bml`

Examples

```
## Not run:
# Multiple-membership: parties (pid) within governments (gid)
id(pid, gid)

# Hierarchical: governments within countries
id(cid)

## End(Not run)
```

log_lik *Pointwise log-likelihood of a bml model*

Description

Pointwise log-likelihood of a bml model

Usage

```
log_lik(object, ...)

## S3 method for class 'bml'
log_lik(object, ...)
```

Arguments

object	A fitted bml model. Available for gaussian and bernoulli families fitted with monitor = "log_lik".
...	Unused.

Value

A draws x observations matrix of pointwise log-likelihood values.

loo.bml *Efficient approximate leave-one-out cross-validation for bml models*

Description

PSIS-LOO via the **loo** package, computed from the retained pointwise log-likelihood. Available for gaussian and bernoulli families fitted with monitor = "log_lik". Compare models with loo::loo_compare().

Usage

```
## S3 method for class 'bml'
loo(x, ...)
```

Arguments

x	A fitted bml model.
...	Passed to loo::loo.matrix().

Value

A loo object.

make_jagscode *Generate the JAGS model code / data without fitting*

Description

The bml analogues of `brms::make_stancode()` / `make_standata()`: build the model exactly as `bml` would, but return the generated JAGS model string (`make_jagscode`) or the JAGS data list (`make_jagsdata`) instead of fitting.

Usage

```
make_jagscode(
  formula,
  data,
  family = stats::gaussian(),
  prior = NULL,
  monitor = "summary"
)

make_jagsdata(
  formula,
  data,
  family = stats::gaussian(),
  prior = NULL,
  monitor = "summary"
)
```

Arguments

`formula`, `data`, `family`, `prior`	As in `bml`.
`monitor`	As in `bml` (affects which nodes the model defines).

Value

`make_jagscode()`: a character scalar of class `"bml_jagscode"` (prints nicely). `make_jagsdata()`: a named list.

Examples

```
## Not run:
code <- make_jagscode(
  y ~ x + mm(id = id(pid, gid), vars = vars(z), w = w(~ 1/n), fn = fn("sum")),
  data = dat, family = gaussian()
)
code

## End(Not run)
```

mcmcDiag *Summarize MCMC convergence diagnostics*

Description

Computes common convergence diagnostics for selected parameters from a JAGS/BUGS fit and returns a compact, report-ready table. The diagnostics include Gelman–Rubin $\hat{R}$, Geweke z-scores, Heidelberger-Welch stationarity p-values, and autocorrelation at a user-specified lag.

Usage

```
mcmcDiag(bml.out, parameters, lag = 50)
```

Arguments

bml.out	A fitted bml model with monitor = "parameters".
parameters	Character vector of parameter names (or patterns) to extract. These may be exact names or patterns (e.g., a prefix like "b" that matches "b[1]", "b[2]", ...).
lag	Integer specifying the lag at which to compute autocorrelation, in units of stored (post-thinning) samples. Default: 50. Lower values (e.g., 10) capture short-range dependence; higher values assess whether the chain has mixed well over longer intervals. If lag is greater than or equal to the number of stored draws per chain (e.g., because n.thin > 1 reduced the number of stored draws), it is automatically lowered with a warning.

Details

Internally, the function converts the retained parameter draws to a coda::mcmc.list, then computes per-chain diagnostics and averages them across chains for each parameter:

- **Gelman–Rubin** ($\hat{R}$): coda::gelman.diag(). Values close to 1 indicate convergence; a common heuristic is $\hat{R} \leq 1.1$.
- **Geweke** z-score: coda::geweke.diag(). Large absolute values (e.g., $|z| > 2$) suggest lack of convergence.
- **Heidelberger–Welch** p-value: coda::heidel.diag() tests the null of stationarity in the chain segment.
- **Autocorrelation**: coda::autocorr() at the specified lag, averaged across chains. Values near zero indicate good mixing; persistent autocorrelation suggests the chain needs thinning or reparameterization.

All statistics are rounded to three decimals. The returned table is transposed so that *rows are diagnostics* and *columns are parameters*.

Value

A data.frame with one row per diagnostic and one column per parameter; cell entries are the average diagnostic values across chains. Row names include: "Gelman/Rubin convergence statistic", "Geweke z-score", "Heidelberger/Welch p-value", "Autocorrelation (lag <lag>)".

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

References

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4), 457-472.

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4), 434-455.

See Also

gelman.diag, geweke.diag, heidel.diag, autocorr

Examples

```
## Not run:
data(coalgov)

# Fit model
m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE),
  family = weibull(),
  monitor = "parameters",
  data = coalgov
)

# Check convergence for main parameters
mcmcDiag(m1, parameters = "b")  # All b coefficients

# Check specific parameters
mcmcDiag(m1, parameters = c("b[1]", "b[2]", "shape"))

# Check mm block parameters
mcmcDiag(m1, parameters = c("b.fn.1", "sigma.mm.1"))

# Custom autocorrelation lag
mcmcDiag(m1, parameters = "b", lag = 100)

# Interpreting results:
# - Gelman-Rubin < 1.1: Good convergence
# - |Geweke z| < 2: No evidence against convergence
# - Heidelberger p > 0.05: Chain appears stationary
# - Low autocorrelation: Good mixing

## End(Not run)
```

mm *Define a multiple-membership block (the micro-to-macro aggregation)*

Description

Specifies a multiple-membership level where group-level units (e.g., occupations) are composed of member-level units (e.g., tasks). Each block aggregates the weighted member records into one or more *group-level features* whose coefficients are estimated by the main model. The components map onto the framework's notation $\theta^{micro,f}(M_{it})$:

- `id = id(mmid, mainid)`: the membership structure S_{it}
- `vars = vars(x)`: the member attributes x_{kt}
- `w = w(~ ...)`: the weights w_{ikt} (who counts)
- `fn = fn("...")`: the aggregation function f (how contributions combine); `fn("sum")` is the additive case, other types are emergent features (variance, concentration, thresholds, ...)
- `RE = re(...)`: member random effects, aggregated by the weights ($\sum_k w_{ikt} u_{0k}$, plus random slopes via `re(1 + x)`)
- `name`: block name for referencing the feature in main-formula interactions (auto-generated when omitted)

Usage

```
mm(
  id,
  vars = NULL,
  w = NULL,
  fn = NULL,
  RE = NULL,
  FE = NULL,
  name = NULL,
  ...
)
```

Arguments

`id`	An `id` object: `id(mmid, mainid)`.
`vars`	A `vars` object with member attributes, or `NULL` for weights-only blocks (RE-only, or `fn("hhi")`-family).
`w`	A `w` object specifying the weights. Default: `w(~ 1/n, scale = TRUE)` (equal weights).
`fn`	A `fn` object selecting the aggregation function. Default: `fn("sum")` (the additive weighted mean).
`RE`	Random effects: `TRUE` (shorthand for `re(1)`), a `re` object, or `NULL`/`FALSE` for none. Only available with `fn("sum")`: member random effects do not compose with dispersion-type features.

FE	Fixed effects: a fe object, or NULL. Mutually exclusive with RE. With many members this is weakly identified — prefer RE (partial pooling) at the member level.
name	Optional block name (unquoted or string) used to reference the block's feature in main-formula interactions (e.g. Ax:education). Auto-generated from the feature when omitted (e.g. A_x, V_x). Must not collide with a data column.
...	Not used; catches removed arguments (c =, and ar =, which moved into the effects grammar: RE = re(1, ar = TRUE)) with a migration message.

Details

One mechanism. Every block emits group-level feature(s) whose coefficients are main-model coefficients (class "b", labeled by the feature name, e.g. A_x). fn("sum") is the additive case $\theta = \beta A_x + \sum_k w_k u_{0k}$: the feature A_x plus the optional RE. There is no separate "sum mode".

Multiple blocks can be combined with + to stack features (mean + variance + concentration, ...). RE can be specified for one block per member-id group.

Value

A bml_mm object.

References

Rosche, B. (2026). A Multilevel Model for Coalition Governments: Uncovering Party-Level Dependencies Within and Between Governments. *Political Analysis*.

See Also

bml, id, vars, w, fn, re, fe, hm

Examples

```
## Not run:
# Additive aggregation (weighted mean effect)
mm(id = id(task, occ), vars = vars(x), w = w(~ importance, scale = TRUE), fn = fn("sum"))

# With member random intercepts
mm(id = id(task, occ), vars = vars(x), w = w(~ 1/n), fn = fn("sum"), RE = TRUE)

# Random intercept + slope (residual effect heterogeneity)
mm(id = id(task, occ), vars = vars(x), w = w(~ 1/n), fn = fn("sum"), RE = re(1 + x))

# Emergent features
mm(id = id(task, occ), vars = vars(x), w = w(~ 1/n), fn = fn("var"))
mm(id = id(task, occ), w = w(~ importance), fn = fn("hhi"))
mm(id = id(task, occ), vars = vars(x), w = w(~ 1/n),
   fn = fn("threshold", c = est(), kappa = 10))
mm(id = id(task, occ), vars = vars(x), w = w(~ 1/n), fn = fn("smax", kappa = est()))

# Named block, referenced in a cross-level interaction
```

```
bml(Y ~ education + Ax:education +
      mm(name = Ax, id = id(task, occ), vars = vars(x), w = w(~ 1/n), fn = fn("sum")),
    data = dat)

## End(Not run)
```

`monetPlot` *Visualize posterior distributions with density and trace plots*

Description

Creates a combined diagnostic plot showing both the posterior density and MCMC trace plot for a specified parameter. Helps assess convergence and visualize posterior uncertainty. The plot displays the median and 90% highest posterior density (HPD) interval.

Usage

```
monetPlot(bml, parameter, label = NULL, r = 2, yaxis = TRUE)
```

Arguments

`bml`	A fitted model object of class `"bml"` returned by `bml`. Must be fitted with `monitor = "parameters"`.
`parameter`	Character string specifying the parameter to plot. Must use the internal parameter name (i.e., row names from `bml$reg.table`). Examples: `"b[1]"` (intercept), `"b[2]"` (first covariate), `"b.fn.1"` (first mm block coefficient), `"sigma.mm"` (mm random effect SD).
`label`	Optional character string for the parameter label displayed on the plot. If `NULL` (default), uses the internal parameter name.
`r`	Number of decimal places for displayed quantiles and statistics. Default: 2.
`yaxis`	Logical; if `TRUE` (default), display axis titles ("Density" and "Scans"). If `FALSE`, omit axis titles for cleaner appearance when combining multiple plots.

Details

Interpreting the Plot:

- **Density panel**: Shows the posterior distribution. The dashed line marks the median (central estimate). Shading indicates the 90% credible region.
- **Trace panel**: Shows parameter values across MCMC iterations for each chain. Good mixing looks like "fuzzy caterpillars" with chains overlapping. Poor mixing shows trends, stickiness, or separation between chains.

Convergence Checks:

- Chains should overlap and explore the same space

- No sustained trends or drift
- Rapid mixing (no long autocorrelation)

Use mcmcDiag for formal convergence statistics (Gelman-Rubin, Geweke, etc.).

Value

A ggplot object (using patchwork) combining two panels:

- **Top panel**: Posterior density with shaded 90% HPD interval. Solid vertical line at zero, dashed line at posterior median.
- **Bottom panel**: Trace plot showing MCMC iterations across chains. Same reference lines as top panel. Helps diagnose convergence and mixing.

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

bml, mcmcDiag, summary.bml

Examples

```
## Not run:
data(coalgov)

# Fit model with parameter draws retained
m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE) +
     hm(id = id(cid)),
  family = weibull(),
  monitor = "parameters",  # Required for monetPlot
  data = coalgov
)

# Plot intercept
monetPlot(m1, parameter = "b[1]", label = "Intercept")

# Plot majority coefficient with custom label
monetPlot(m1, parameter = "b[2]", label = "Majority Government Effect")

# Plot mm coefficient
monetPlot(m1, parameter = "b.fn.1", label = "Party Fragmentation")

# Plot random effect SD
monetPlot(m1, parameter = "sigma.mm.1")

# List available parameters
rownames(m1$reg.table)
```

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

`posterior_predict` *Draws from the posterior predictive distribution*

Description

Returns draws of replicated outcomes `y_rep` (the monitored `pred` node). `newdata` is not supported yet: multiple-membership prediction for new units requires the new units' membership map, which is future work.

Usage

```
posterior_predict(object, ...)

## S3 method for class 'bml'
posterior_predict(object, ndraws = NULL, ...)
```

Arguments

`object`	A fitted `bml` model with `monitor = "predictive"`.
`...`	Unused.
`ndraws`	Optional number of draws to return (subsampled without replacement).

Value

A draws x observations matrix.

`pp_check` *Posterior predictive checks for bml models*

Description

Graphical posterior predictive checks via **bayesplot** (must be installed). Compares the observed outcome with replicated outcomes from `posterior_predict.bml`.

Usage

```
pp_check(object, ...)

## S3 method for class 'bml'
pp_check(object, type = NULL, ndraws = 30, ...)
```

Arguments

object	A fitted bml model (gaussian or bernoulli, fitted with monitor = "predictive").
...	Passed to the bayesplot function.
type	A **bayesplot** PPC type (without the "ppc_" prefix), e.g. "dens_overlay" (default for gaussian), "bars" (default for bernoulli), "hist", "stat".
ndraws	Number of replicated datasets to plot. Default: 30.

Value

A ggplot object.

---

predict.bml *Predictions from a bml model*

Description

Posterior predictive summaries (one row per observation). For the raw draws use posterior_predict.bml; for the linear predictor use fitted.bml.

Usage

```
## S3 method for class 'bml'
predict(object, summary = TRUE, ...)
```

Arguments

object	A fitted bml model with monitor = "predictive".
summary	Logical; if TRUE (default), return a summary matrix, else the draws matrix.
...	Unused (newdata is not supported yet).

Value

A matrix of posterior predictive summaries, or a draws matrix.

prior *Structured prior specifications*

Description

Specify priors brms-style: a distribution call plus a parameter `class`, optionally narrowed to a single coefficient (`coef`) or block (`group`). Distributions are written on the natural (standard-deviation) scale and translated to JAGS's precision parameterization automatically.

```
prior(normal(0, 10),    class = "b")                    # all fixed coefficients
prior(normal(0, 1),     class = "b", coef = "education")  # one coefficient
prior(exponential(1),   class = "sd")                   # RE standard deviations
prior(lkj(2),        class = "cor")              # RE correlation (re(cor=TRUE))
prior(student_t(3,0,5), class = "Intercept")
prior(normal(0, 1),    class = "w")                   # weight-model parameters
prior(uniform(0, 1),    class = "fn", coef = "c")         # fn shape parameters
```

Combine several with `c()` and pass as `bml(..., prior = c(...))`. Raw JAGS strings remain a power-user escape hatch: character elements such as `"b.fn.1[1] ~ dnorm(0, 0.1)"` are passed through untranslated.

Usage

```
prior(dist, class = "b", coef = "", group = "")
```

Arguments

`dist`	A distribution call. Supported: `normal(mu, sigma)`, `student_t(nu, mu, sigma)`, `cauchy(mu, sigma)`, `exponential(rate)`, `gamma(shape, rate)`, `uniform(lower,upper)`, `lkj(eta)` (class `"cor"` only). Scale parameters are standard deviations, not precisions.
`class`	Parameter class. One of: • `"Intercept"`: the main intercept • `"b"`: fixed coefficients (main-formula terms and block features); narrow with `coef` (term label, e.g. `"education"` or a feature name like `"A_x"`) • `"sd"`: random-effect standard deviations; narrow with `group` (`"mm"`/`"hm"` or a block id like `"mm.1"`) and `coef` (`"Intercept"` or a slope variable) • `"cor"`: the RE intercept-slope correlation (only exists when the user opts into `re(..., cor = TRUE)`) — use `lkj(eta)` • `"sigma"`: the Gaussian residual SD • `"shape"`: the Weibull shape parameter • `"w"`: weight-model parameters (free symbols in `w()`) • `"fn"`: `fn()` shape parameters (`c`, `kappa`, `p`, DSL free symbols); narrow with `coef`
`coef`	Optional coefficient/parameter name to narrow the prior to.
`group`	Optional block identifier (e.g. `"mm.1"`, `"hm.2"`, or just `"mm"`/`"hm"`).

Value

A bml_prior data frame (one row per prior); combine with c().

See Also

get_prior, bml

---

ranef.bml *Extract random effects from a bml model*

Description

Returns the posterior means of the unit-level random effects: member effects per mm() member-id group (intercepts, and slopes when re(1 + x) was used) and unit effects per hm() block.

Usage

```
## S3 method for class 'bml'
ranef(object, ...)
```

Arguments

object	A fitted bml model with monitor = "random_effects".
...	Unused.

Value

A list with elements mm and hm.

---

re *Random effects: partial pooling*

Description

Specifies the random-effect structure of an mm or hm block, mirroring the left-hand side of an lme4/brms (... | group) term. The grouping is implied by the block's id() and therefore not repeated.

```
RE = TRUE            # shorthand for re(1)
re(1)                # random intercept
re(1 + x)            # intercept + slope on x (INDEPENDENT, the default)
re(1 + x, cor = TRUE)# intercept + slope, correlated (lme4/brms `|`) - opt-in
re(0 + x)            # slope only (also re(x - 1))
re(1, ar = TRUE)     # random walk over participation order
re(1, ar = year)     # random walk over calendar time (gap-scaled variance)
```

The default cor = FALSE deliberately diverges from brms's correlated default: the correlated draw is a real JAGS cost, and in mm() blocks the member random effects are only read through the weighted aggregate sum_k w_k u_k, so the correlation is rarely of interest and often weakly identified. Opt in with cor = TRUE (supported for a single slope, i.e. the 2x2 case).

Usage

```
re(x = 1, cor = FALSE, ar = FALSE)
```

Arguments

x	The effects expression: 1, 1 + var, 0 + var, or var - 1. For mm() blocks, slope variables are member-level columns; for hm() blocks they are main-level columns varying within the nesting units.
cor	Logical; if TRUE, intercept and slope get a joint bivariate-normal prior (correlated). Default FALSE (independent).
ar	Autoregressive random walk for the intercept across a unit's repeated participations/observations. FALSE (default): independent effects. TRUE: random walk over participation order, $u_t \sim N(u_{t-1}, \sigma^2)$. An unquoted column name (e.g. ar = year): random walk over that (numeric) time variable, with the step variance scaled by the normalized time gaps, $u_t \sim N(u_{t-1}, \sigma^2\,\Delta_t/\bar{\Delta})$ — equal spacing reproduces the ar = TRUE model, and σ keeps its per-average-step interpretation. Requires an intercept-only re(1); time values must be unique within each unit.

Value

A bml_re object.

See Also

fe, mm, hm

---

summary.bml *Summarize a fitted bml model*

Description

S3 method for summarizing bml model objects. Returns a formatted table of parameter estimates with posterior means, standard deviations, and credible intervals, along with model information and convergence statistics.

Usage

```
## S3 method for class 'bml'
summary(object, r = 3, diagnostics = TRUE, ...)
```

Arguments

object	A fitted model object of class "bml" returned by bml.
r	Number of decimal places for rounding numeric output. Default: 3.
diagnostics	Logical; include per-parameter R-hat and effective sample sizes in the printed table? Default: TRUE.
...	Additional arguments (currently unused).

Details

The summary method rounds all numeric values for readability while preserving the underlying structure and metadata from the fitted model. All columns remain accessible via standard data frame indexing (e.g., $Parameter, $Estimate).

For Cox models with piecewise baseline hazards (when cox_intervals is specified), the outcome description includes the number of intervals used.

Value

A data frame of class "bml_summary" containing rounded parameter estimates with the following columns:

- Parameter: Labeled parameter names
- Estimate: Posterior mean
- Est.Error: Posterior standard deviation
- Q2.5/Q97.5: 95% credible interval
- Rhat, ESS_bulk, and ESS_tail: convergence diagnostics when diagnostics = TRUE

The object includes metadata attributes printed above the table:

- Outcome family and link function
- Estimate type (posterior mean from MCMC)
- Credible interval specification (95% equal-tailed)
- Level specification (mm and hm block details)
- DIC (Deviance Information Criterion) for model comparison

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

bml, tidy.bml, coefPlot, monetPlot, mcmcDiag

Examples

```
## Not run:
data(coalgov)

# Fit model
m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE) +
     hm(id = id(cid)),
  family = weibull(),
  data = coalgov
)

# View summary
summary(m1)

# Summary with more decimal places
summary(m1, r = 4)

# Access specific columns
s <- summary(m1)
s$Parameter  # Parameter names
s$Estimate   # Posterior means
s$Q2.5       # Lower credible bounds

# Custom posterior summaries (requires monitor = "parameters")
draws <- posterior::as_draws_df(m1)

# Select specific parameters and compute custom summaries
draws |>
  dplyr::select(dplyr::starts_with("b[")) |>
  tidyr::pivot_longer(everything(), names_to = "param") |>
  dplyr::group_by(param) |>
  dplyr::summarise(
    median = median(value),
    mad    = mad(value),
    q05    = quantile(value, 0.05),
    q95    = quantile(value, 0.95)
  )

## End(Not run)
```

`tidy.bml` *Tidy a fitted bml model*

Description

Turns a fitted `bml` model into a tidy tibble following the **broom** conventions: one row per parameter with columns `term`, `estimate`, `std.error`, `conf.low`, and `conf.high`. Because the model

is Bayesian, `estimate` is the posterior mean, `std.error` the posterior standard deviation, and `conf.low`/`conf.high` the bounds of an equal-tailed credible interval.

With these methods, ecosystem tools that build on **broom** work out of the box, e.g. `modelsummary::modelsummary(list(m1 m2))` for side-by-side regression tables or `dotwhisker::dwplot(m1)` for coefficient plots.

Usage

```
## S3 method for class 'bml'
tidy(
  x,
  conf.int = TRUE,
  conf.level = 0.95,
  component = c("all", "fixed", "random", "weights", "fn"),
  ...
)
```

Arguments

`x`	A fitted model object of class `"bml"` returned by `bml`.
`conf.int`	Logical. Include credible interval columns `conf.low`/`conf.high`? Default: `TRUE`.
`conf.level`	Width of the equal-tailed credible interval. Default: 0.95, which is read directly from the fitted model. Other levels are computed from the posterior draws and therefore require `monitor = "parameters"`.
`component`	Which parameters to return: • `"all"` (default): all of the below • `"fixed"`: all class-`"b"` coefficients (main-formula terms, block features, interactions) • `"random"`: variance parameters (`sd(...)`, `cor(...)`, `sigma`, Weibull shape) • `"weights"`: weight-model parameters (`w[...]`) • `"fn"`: aggregation-function shape parameters (`fn[...]`)
`...`	Additional arguments (currently unused).

Value

A `tibble` with columns `term`, `estimate`, `std.error`, `conf.low`, `conf.high` (if `conf.int`), and `component`.

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

`bml`, `glance.bml`, `bmlCompare`, `coefPlot`, `summary.bml`

Examples

```
## Not run:
data(coalgov)

m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE),
  family = weibull(),
  data = coalgov
)

tidy(m1)                      # regression coefficients
tidy(m1, component = "all")   # including weight and variance parameters
tidy(m1, conf.level = 0.9)    # requires monitor = "parameters"

# Multi-model comparison table (see also bmlCompare()):
models <- list(base = m1, weighted = m2)
purrr::imap_dfr(models, \(m, name)
  tidy(m) |>
    dplyr::mutate(model = name, N = glance(m)$nobs, DIC = glance(m)$DIC)
)

## End(Not run)
```

`varDecomp` *Variance decomposition for fitted bml models*

Description

Computes a posterior variance decomposition and intraclass correlation coefficients (ICCs) from a fitted `bml` model. The function automatically discovers all variance components (sigma parameters) in the model, applies weight adjustments for multiple-membership levels, and returns posterior summaries.

Usage

```
varDecomp(model, uncertainty = "sd", r = 2)
```

Arguments

`model`	A fitted model object of class `"bml"` returned by `bml`. Must have been fitted with `monitor = "parameters"`.
`uncertainty`	Uncertainty measure to report. One of `"sd"` (posterior standard deviation, the default), `"mad"` (median absolute deviation), or `"ci"` (95% credible interval with lower/upper bounds).
`r`	Number of decimal places for rounding numeric output. Default: 2.

Details

Variance decomposition. The total variance of the outcome is partitioned into additive components, one for each level in the model:

$$\mathrm{Var}(y) = \sigma_1^2 + \sigma_2^2 + \ldots$$

Each component contributes variance σ^2, except for multiple-membership (MM) levels, where the effective variance contribution is scaled by the average of the summed squared weights across groups.

$$\mathrm{Var}_{\mathrm{mm}} = \sigma^2_{\mathrm{mm}} \cdot \overline{w^2}, \quad \overline{w^2} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k} w_{ik}^2$$

This weight adjustment accounts for the fact that the member-level variance is distributed across multiple members with potentially unequal influence. With equal weights ($w_{ik} = 1/n_i$), the effective variance shrinks as group size increases.

Intraclass Correlation Coefficient (ICC). The ICC for a given level is the proportion of total variance attributable to that level:

$$\rho_l = \frac{\sigma_l^2}{\sum_{l'} \sigma_{l'}^2}$$

Intuitively, the ICC answers: "What fraction of the total variation in the outcome is due to differences between units at this level?" An ICC of 0.30 for the country level, for example, means that 30% of the outcome variation can be attributed to between-country differences.

ICCs are computed per posterior draw and then summarized, properly propagating uncertainty from the MCMC samples.

Family-specific handling:

- **Gaussian / Weibull**: The residual `sigma` from the model is used directly.
- **Binomial**: There is no residual sigma. The latent logistic residual variance $\pi^2/3 \approx 3.29$ is used instead.
- **Cox**: There is no residual variance. ICCs are computed among the non-residual components only.

Value

A data frame of class `"bml_varDecomp"` with one row per variance component. Always includes `Component`, `sigma`, and `ICC` columns. Additional columns depend on `uncertainty`:

- `"sd"`: `sigma_sd` and `ICC_sd`
- `"mad"`: `sigma_mad` and `ICC_mad`
- `"ci"`: `sigma_lb`, `sigma_ub`, `ICC_lb`, `ICC_ub`

Author(s)

Benjamin Rosche <benrosche@nyu.edu>

See Also

bml, summary.bml

Examples

```
## Not run:
data(coalgov)

m1 <- bml(
  Surv(dur_wkb, event_wkb) ~ 1 + majority +
    mm(id = id(pid, gid), vars = vars(cohesion), w = w(~ 1/n), fn = fn("sum"), RE = TRUE) +
     hm(id = id(cid)),
  family = weibull(),
  data = coalgov
)

varDecomp(m1)
varDecomp(m1, uncertainty = "ci")
varDecomp(m1, uncertainty = "mad")

## End(Not run)
```

---

vars *Specify member-level attributes for mm() blocks*

Description

Helper function used within mm to specify the member-level attributes (the effect covariates x_{kt}) that the block's aggregation function fn reads. Supports both free variables (with coefficients estimated by the main model) and fixed variables (coefficients held constant via fix).

Usage

```
vars(...)
```

Arguments

...	Unquoted variable names, combined with + (formula-style) or as separate arguments. Supports: • Simple variables: vars(x + y) or vars(x, y) • Interactions: vars(x * y) (main effects + interaction) or vars(x:y) (interaction only; the member-paired interaction $H_{xz} = \sum_k w_k x_k z_k$) • Transformations: vars(I(x^2)) • Fixed coefficients: vars(fix(x, 1.0) + y) Note: Numeric literals like 1, 0, or -1 are ignored (no intercept support in mm blocks).

Value

A bml_vars object (list with formula, free, fixed), or NULL if empty.

See Also

fix, mm, fn

Examples

```
## Not run:
vars(income)
vars(income + education)
vars(income:education)      # member-paired interaction only
vars(x, z)                  # two attributes (e.g. for fn("cov"))
vars(fix(exposure, 1.0) + income)

## End(Not run)
```

---

w *Specify the membership weights of an mm() block*

Description

Defines the weights w_{ikt} — how much each member counts in the block's aggregation. w() defines a within-group *measure* over members; the aggregation function fn chooses which function of that measure (and of the vars attributes) to read off. Weights can be fixed, taken from observed data, or estimated from member characteristics:

```
w(~ 1/n)                          # equal weights
w(~ importance, scale = TRUE)     # observed, normalized
w(~ ilogit(b0 + b1 * q))          # estimated: b0, b1 are free parameters
```

One parameter rule (shared with fn): any symbol in the formula that is not a data column or reserved word (n, the group size) is a *free parameter* with a default prior. The build always messages the detected parameters (e.g. Estimating parameters: b0, b1 (w)), so a misspelled column name shows up immediately. Priors are set via prior(..., class = "w").

Who counts vs. how contributions combine: parameters that act on *external* member characteristics (importance, exposure time) belong in w(); parameters that act on the *effect attribute* (whatever is in vars()) belong in fn. Building weights from the same variable that is being aggregated (e.g. w(~ exp(om * x)) with vars(x)) is near-equivalent to a smooth-max in fn() and is flagged with a warning.

Usage

```
w(formula = ~1/n, scale = TRUE)
```

Arguments

formula	One-sided formula defining the raw weights. Group-level aggregation shortcuts (min(x), max(x), mean(x), sum(x), sd(x), var(x), median(x), mode(x), range(x), first(x), last(x), quantile(x, p)) are pre-computed in R. JAGS math functions (exp, log, ilogit, ...) pass through.
scale	Logical; if TRUE (default), weights are normalized to sum to 1 within each group, making them a probability measure over the group's members. Emergent fn() types require scale = TRUE.

Details

Weight functions with estimated parameters must produce bounded, positive values across plausible parameter values; unbounded weight functions can crash the sampler. Use bounded forms such as w(~ ilogit(b0 + b1 * q), scale = TRUE) or the generalized logistic w(~ 1 / (1 + (n - 1) * exp(-(b0 + b1 * q))), scale = TRUE), keep scale = TRUE, standardize covariates, and use informative priors (prior(normal(0, 1), class = "w")). Weight parameters are initialized at 0 for stable starting values.

Value

A bml_w object.

References

Rosche, B. (2026). A Multilevel Model for Coalition Governments: Uncovering Party-Level Dependencies Within and Between Governments. *Political Analysis*.

See Also

mm, fn, bml

Examples

```
## Not run:
w(~ 1/n)                                # equal weights
w(~ importance, scale = TRUE)           # observed weights, normalized
w(~ ilogit(b0 + b1 * seniority))        # estimated weights
w(~ duration, scale = FALSE)            # unnormalized (aggregating sums)

## End(Not run)
```

`waic.bml` *Widely applicable information criterion for bml models*

Description

Widely applicable information criterion for bml models

Usage

```
## S3 method for class 'bml'
waic(x, ...)
```

Arguments

`x`	A fitted `bml` model (see `loo.bml` for availability).
`...`	Passed to `loo::waic.matrix()`.

Value

A `waic` object.

Index

∗ **datasets**
coalgov, 12

as_draws.bml, 3
as_draws_array.bml *(*as_draws.bml*)*, 3
as_draws_df.bml *(*as_draws.bml*)*, 3
as_draws_matrix.bml *(*as_draws.bml*)*, 3
autocorr, *30*

bernoulli, *4*
bernoulli *(*bml-families*)*, 7
bml, 3, *7, 8*, *10, 11*, *14*, *16, 17*, *23–26*, *28*, *32–34*, *38*, *40–43*, *45*, *47*
bml-families, 7
bml_delta, 8, *9, 10*
bml_realloc, *8*, 9, *10*
bml_turnover, *8*, 9, *9*
bmlCompare, *6*, 10, *17*, *24*, *42*

coalgov, 12
coef.bml, 15
coefPlot, *11*, 16, *40*, *42*
conditional_effects, 17
cox, *4*
cox *(*bml-families*)*, 7

est, 18, *22*

fe, *3*, *6*, 19, *25*, *32*, *39*
fitted.bml, 19, *36*
fix, 20, *45, 46*
fixef.bml, *6*, 21
fn, *3*, *6*, *18*, 21, *31, 32*, *45–47*

gelman.diag, *30*
get_prior, *4*, *6*, 22, *38*
geweke.diag, *30*
glance.bml, *10, 11*, 23, *42*
glm, *7*

heidel.diag, *30*
hm, *5, 6*, *19*, 24, *26*, *32*, *38, 39*

id, *25*, 26, *31, 32*

log_lik, 27
loo.bml, *6*, 27, *48*

make_jagscode, *5, 6*, 28
make_jagsdata *(*make_jagscode*)*, 28
mcmcDiag, 29, *34*, *40*
mm, *5, 6*, *19, 20*, *22*, *25, 26*, 31, *38, 39*, *45–47*
monetPlot, *17*, 33, *40*

posterior_predict, 35
posterior_predict.bml, *35, 36*
pp_check, 35
predict.bml, 36
prior, *4*, *6*, *18*, *22, 23*, 37

ranef.bml, *6*, 38
re, *3*, *6*, *19*, *25*, *31, 32*, 38

summary.bml, *6*, *21*, *34*, 39, *42*, *45*

tidy.bml, *10, 11*, *16, 17*, *24*, *40*, 41

varDecomp, *6*, 43
vars, *20*, *31, 32*, 45, *46*

w, *3*, *6*, *22*, *31, 32*, 46
waic.bml, 48
weibull, *4*
weibull *(*bml-families*)*, 7