

Package: lmebayes (via r-universe)

October 2, 2026

Type Package

Title Bayesian Linear Mixed-Effects Models via Two-Block Gibbs Sampling

Version 0.1.0

Date 2026-08-10

Description Provides Bayesian linear and generalized linear mixed-effects model fitting with near-independent posterior samples. The main functions mirror the 'lme4' lmer() and glmer() interfaces (X/beta for fixed effects, Z/b for random effects, D for the random-effects covariance) while adding prior family specifications for Gaussian, Poisson, binomial, and Gamma models with log-concave likelihoods. Sampling uses efficient two-block Gibbs sampling procedures with parallel chains and theoretically driven stopping rules. Gibbs sampling engines are provided by the 'lmebayesCore' package.

License GPL (>= 2)

URL <https://github.com/knygren/lmebayes>

BugReports <https://github.com/knygren/lmebayes/issues>

Imports glmbayesCore (>= 0.6.0), lmebayesCore, glmbayes, stats, lme4, Rcpp (>= 1.1.1), RcppParallel, Rdpack (>= 0.11-0), opencltools (>= 0.8.0)

RdMacros Rdpack

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Depends MASS, R (>= 3.5.0)

Suggests coda, ggplot2, bayesrules, bayestestR, glmmTMB, LearnBayes, testthat (>= 3.0.0), spelling, knitr, rmarkdown

VignetteBuilder knitr

SystemRequirements Optional OpenCL support. If available, GPU acceleration will be used; otherwise, computation runs on CPU.

Encoding UTF-8

Roxygen list(markdown = TRUE)
RoxygenNote 7.3.3
Config/testthat/edition 3
LazyData true
Language en-US
Config/pak/sysreqs cmake make libicu-dev ocl-icd-opencl-dev
Repository <https://knygren.r-universe.dev>
Date/Publication 2026-10-02 21:37:46 UTC
RemoteUrl <https://github.com/knygren/lmebayes>
RemoteRef HEAD
RemoteSha 9447cd81eadcaf1bf4ccfead81f07ba915de2c2d

Contents

lmebayes-package	3
AMI	5
as.data.frame.ranef.lmerb	6
BikeSharing	6
Boston_centered	8
carinsca	9
Cleveland	10
coef.lmerb	11
dGamma_list	12
directional_tail	14
fixef.lmerb	15
glmb	16
glmbBlock	23
glmerb	25
has_opencl	31
lmb	31
lmbBlock	37
lmerb	40
model_setup	47
pfamily	52
pfamily_list	56
plot_mean_convergence	58
plot_var_convergence	59
print.ranef.lmerb	61
Prior_Setup	62
Prior_Setup_GLMM	67
Prior_SetupGroup	77
ranef.lmerb	80
rglmerb	81
rlmerb	84

summary.bglmb	88
summary.blmb	89
summary.lmerb	89
summary.ranef.lmerb	91
summary_sigma2	91

Index	93
--------------	-----------

lmebayes-package	<i>lmebayes: Bayesian Linear Mixed-Effects Models via Two-Block Gibbs Sampling</i>
------------------	--

Description

Two-block Gibbs samplers for Bayesian linear and generalized linear mixed-effects models, following **lme4** notation. Builds on **glmbayes** for iid GLM sampling within blocks.

Details

Row-block interfaces include **lmbBlock** and **glmbBlock**; mixed-model setup from **lme4** formulas via **model_setup**. Lower-level simulation uses **simfunction** and envelope utilities from **lmebayesCore**.

See the package README at <https://github.com/knygren/lmebayes> for examples.

Vignette index

The vignettes are a curriculum, meant to be read roughly in order. Each assumes the previous ones.

`vignette("Chapter-00")` Roadmap: what the package does and which chapter answers which question.

`vignette("Chapter-01")` Getting started: one model end to end on `lme4::sleepstudy`.

`vignette("Chapter-02")` From `lm/glm` to mixed models: pooling, the centered parameterization, and the model-class restrictions.

`vignette("Chapter-03")` The prior workflow: `model_setup`, `Prior_Setup_GLMM`, `pfamily_list`.

`vignette("Chapter-04")` Linear mixed models with group-level covariates and cross-level moderation.

`vignette("Chapter-05")` Reading the output: every accessor and container.

`vignette("Chapter-06")` Convergence and `tv_tol`.

`vignette("Chapter-07")` Estimated variance components and the pilot stage.

`vignette("Chapter-08")` Observation dispersion, including per-group σ_j^2 .

`vignette("Chapter-09")` Poisson GLMMs.

`vignette("Chapter-10")` Binomial GLMMs.

`vignette("Chapter-11")` Point estimates without simulation.

`vignette("Chapter-12")` Row-block models.

`vignette("Chapter-13")` Comparison with **lme4** and **glmmTMB**.

Engine internals, derivations, and convergence theory are documented in the Chapter-B* vignettes of **lmebayesCore**; the iid GLM stack underneath is documented in **glmbayes**.

OpenCL startup checks

In interactive sessions, attaching the package with `library(lmebayes)` may emit a short `packageStartupMessage` when `has_opengl()` is `FALSE` (typical for CRAN binaries) but a GPU or OpenCL stack appears available on the host. OpenCL modelling paths require a source install with OpenCL at compile time; `has_opengl()` then reports whether that build succeeded. Set `options(glmbayes.quiet_opengl_startup = TRUE)` to suppress attach notes (recommended for CI and R CMD check).

Author(s)

Kjell Nygren

References

There are no references for Rd macro `\insertAllCites` on this help page.

See Also

`lmerb`, `model_setup`, `lmbBlock`, `glmbBlock`; `simfunction`, `EnvelopeBuild`; `lmb` and `glmb` for fixed-effects-only Bayesian linear and generalized linear models; `rlmb` and `rglmb` for iid posterior draws.

Useful links:

- GitHub: <https://github.com/knygren/lmebayes>

Examples

```
set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
print(d.AD <- data.frame(treatment, outcome, counts))

glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())

## Plain iid GLM fitting engine (rglmb) is not re-exported from lmebayes;
## it lives in glmbayesCore, so we source it from there directly.
ps <- Prior_Setup(counts ~ outcome + treatment, family = poisson())

rglmb.D93 <- glmbayesCore::rglmb(
  n = 200,
  y = ps$y,
  x = as.matrix(ps$x),
  pfamily = dNormal(mu = ps$mu, Sigma = ps$Sigma),
  family = poisson(),
  weights = rep(1, nrow(ps$x))
)
print(rglmb.D93)
summary(rglmb.D93)
```

AMI	<i>Amitriptyline overdose data</i>
-----	------------------------------------

Description

Data with information on 17 overdoses of the drug amitriptyline

Usage

```
data(AMI)
```

Format

This data frame contains the following columns:

TOT total TCAD plasma level

AMI amount of amitriptyline present in the TCAD plasma level

GEN gender (male = 0, female = 1)

AMT amount of drug taken at time of overdose

PR PR wave measurement

DIAP diastolic blood pressure

QRS QRS wave measurement

Details

Each row is one overdose episode. Variables include total tricyclic antidepressant level, amitriptyline component, gender, reported amount ingested, and ECG-related measures (PR interval, QRS duration, diastolic blood pressure). The dataset is used in package examples for binomial and related regression; see (Dobson 1990) for analogous generalized linear modelling of clinical outcomes.

References

Dobson A~J (1990). *An Introduction to Generalized Linear Models*. Chapman and Hall, London.

Examples

```
##### Start of AMI dataset example #####
```

```
data(AMI)
summary(AMI)
```

```
#####
## End of AMI dataset example
#####
```

<code>as.data.frame.ranef.lmerb</code>
<i>Coerce ranef.lmerb objects to long data.frame format</i>

Description

Coerce ranef.lmerb objects to long data.frame format

Usage

```
## S3 method for class 'ranef.lmerb'
as.data.frame(x, ...)
```

Arguments

- `x` An object of class "ranef.lmerb".
- `...` Ignored.

Value

A data.frame with columns grpvar, term, grp, condval, and condsd (when postVar is available on x).

See Also

[ranef.](#)

BikeSharing	<i>Bike Sharing Dataset (Processed)</i>
-------------	---

Description

A processed version of the UCI Bike Sharing Dataset (hourly data). The data include derived variables for part of day, quarter, and Fourier terms for cyclic effects of hour and month.

Usage

BikeSharing

Format

A data frame with 17,379 observations and 25 variables (original plus derived):

instant Record index.

dteday Date.

season Season (1: spring, 2: summer, 3: fall, 4: winter).

yr Year (0: 2011, 1: 2012).

mnth Month (1–12).

hr Hour (0–23).

holiday Whether the day is a holiday (0/1).

weekday Day of week (0–6).

workingday Working day (0/1).

weathersit Weather situation (1–4).

temp Normalized temperature.

atemp Normalized feeling temperature.

hum Normalized humidity.

windspeed Normalized wind speed.

casual Count of casual users.

registered Count of registered users.

cnt Total count (casual + registered).

hr_num Hour as numeric (same as hr).

month_num Month as integer.

part_of_day Factor: Night (0–5h), Morning (6–11h), Afternoon (12–17h), Evening (18–23h).

quarter Factor: Q1–Q4.

hr_sin Sine term for 24-hour cycle.

hr_cos Cosine term for 24-hour cycle.

mon_sin Sine term for 12-month cycle.

mon_cos Cosine term for 12-month cycle.

Source

UCI Machine Learning Repository: Bike Sharing Dataset. <https://archive.ics.uci.edu/dataset/275/bike+sharing+dataset>

Examples

```
##### Start of BikeSharing dataset example #####
```

```
data("BikeSharing")
head(BikeSharing)
dim(BikeSharing)
```

```
#####
## End of BikeSharing dataset example
#####
```

 Boston_centered

Boston housing data with mean-centered predictors

Description

A copy of [Boston](#) where all predictors (every column except medv) have been mean-centered (subtract column means, no scaling).

Usage

```
data("Boston_centered")
```

Format

A data frame with 506 observations and 14 variables (same names as [Boston](#)). See ?MASS::Boston for variable descriptions.

Source

Derived from MASS::Boston. Original data described in Harrison and Rubinfeld (1978); see ?Boston in MASS.

Examples

```
##### Boston_centered dataset example #####

data("Boston_centered")
head(Boston_centered)
summary(Boston_centered)

## Predictors are mean-centered (column means ~0)
predictors <- setdiff(names(Boston_centered), "medv")
colMeans(Boston_centered[predictors])

form <- medv ~
  crim + zn +
  indus + chas + nox + age + dis + rad + tax + ptratio + black + lstat + rm

lm.boston <- lm(form, data = Boston_centered, x = TRUE, y = TRUE)
summary(lm.boston)

#####
## End of Boston_centered dataset example
#####
```

carinsca

*Canadian Automobile Insurance Claims for 1957-1958***Description**

The data give the Canadian automobile insurance experience for policy years 1956 and 1957 as of June 30, 1959. The data includes virtually every insurance company operating in Canada and was collated by the Statistical Agency (Canadian Underwriters' Association - Statistical Department) acting under instructions from the Superintendent of Insurance. The data given here is for private passenger automobile liability for non-farmers for all of Canada excluding Saskatchewan.

Usage

```
data(carinsca)
```

Format

A data frame with 20 observations on the following 6 variables:

Merit Merit Rating:

- 3 - licensed and accident free 3 or more years
- 2 - licensed and accident free 2 years
- 1 - licensed and accident free 1 year
- 0 - all others

Class 1 - pleasure, no male operator under 25

- 2 - pleasure, non-principal male operator under 25
- 3 - business use
- 4 - unmarried owner or principal operator under 25
- 5 - married owner or principal operator under 25

Insured Earned car years

Premium Earned premium in 1000's

(adjusted to what the premium would have been had all cars been written at 01 rates)

Claims Number of claims

Cost Total cost of the claim in 1000's of dollars

Details

One could apply Poisson regression to the number of claims and gamma regression to the cost per claim.

Source

Bailey, R. A., and Simon, LeRoy J. (1960). Two studies in automobile insurance ratemaking. ASTIN Bulletin, 192-217.

References

Data downloaded from <http://www.statsci.org/data/general/carinsca.html>. That site also contains classical Poisson and Gamma regression examples.

Examples

```
##### Start of carinsca dataset example #####

data(carinsca)
str(carinsca)
head(carinsca)

#####
## End of carinsca dataset example
#####
```

Cleveland

Cleveland Heart Disease Dataset

Description

A cleaned version of the Cleveland heart disease dataset from the UCI Machine Learning Repository. This version contains only complete cases and includes a derived binary outcome variable `hd` indicating the presence ("Yes") or absence ("No") of heart disease.

Usage

```
data("Cleveland")
```

Format

A data frame with 297 observations and 15 variables:

age Age in years (numeric).
sex Sex (0 = female, 1 = male).
cp Chest pain type (numeric code 1–4).
trestbps Resting blood pressure (mm Hg).
chol Serum cholesterol (mg/dl).
fbs Fasting blood sugar > 120 mg/dl (1 = true, 0 = false).
restecg Resting electrocardiographic results (numeric code).
thalach Maximum heart rate achieved.
exang Exercise-induced angina (1 = yes, 0 = no).
oldpeak ST depression induced by exercise relative to rest.
slope Slope of the peak exercise ST segment.
ca Number of major vessels colored by fluoroscopy (0–3).

thal Thalassemia status (numeric code).
num Original UCI disease score (0–4).
hd Binary heart disease indicator: "No" (num = 0) or "Yes" (num > 0).

Source

UCI Machine Learning Repository: Heart Disease Data Set. <https://archive.ics.uci.edu/dataset/45/heart+disease>

Examples

```
##### Start of Cleveland dataset example #####

data("Cleveland")
head(Cleveland)
summary(Cleveland)

#####
## End of Cleveland dataset example
#####
```

coef.lmerb	<i>Extract combined coefficients (fixed + random) by group</i>
------------	--

Description

Mirrors `coef.merMod`: Block~1 stored coefficients (`ranef.lmerb`) are already on the per-group coefficient scale used internally by the sampler (not lme4-style deviations).

Usage

```
## S3 method for class 'lmerb'
coef(object, type = c("mean", "mode"), ...)

## S3 method for class 'glmerb'
coef(object, type = c("mean", "mode"), ...)
```

Arguments

<code>object</code>	An <code>lmerb</code> or <code>glmerb</code> object.
<code>type</code>	"mean" (default when draws are stored) or "mode" (ICM / <code>groupef.mode</code>). When <code>simulate = FALSE</code> , "mean" falls back to "mode" with a warning.
<code>...</code>	Ignored.

Value

A named list (one element per grouping factor) of data.frames with the same layout as `coef(merMod)`.

See Also

[coef.merMod](#), [ranef.lmerb](#).

dGamma_list	<i>Build a named list of dGamma measurement-dispersion priors</i>
-------------	---

Description

Generic for constructing a named list of [dGamma](#) prior objects from a prior-specification object. For mixed models, one [dGamma](#) per group level supplies observation-level σ_j^2 priors for `lmerb(..., group.dispersion = ...)`.

Usage

```
dGamma_list(object, ...)
```

Arguments

<code>object</code>	A prior-specification object.
<code>...</code>	Additional arguments passed to methods (e.g. <code>max_disp_perc_measurement</code>). For <code>print.dGamma_list()</code> , passed to <code>print.pfamily</code> .

Details

Where [pfamily_list](#) configures the *between*-group variance components τ_k^2 , `dGamma_list()` configures the *within*-group observation variance. It returns one [dGamma](#) prior per group level, so each group gets its own σ_j^2 rather than sharing a pooled value.

Each element places a Gamma prior on the *precision* $1/\sigma_j^2$ (hence `Inv_Dispersion = TRUE`) and carries a truncation window `disp_lower/disp_upper` calibrated from that group's own data. Because groups differ in size and in residual scatter, the windows differ across groups – often by an order of magnitude – which is the point of the per-group treatment.

For [Prior_Setup_GLMM](#) with a per-group `dispformula`, each group's `dGamma()` uses `shape_ING/rate` from `object$group.ing_prior` (Part VI calibration; see `inst/DGAMMA_LIST_MARGINAL_AND_BOUNDS.md`). Truncation bounds are posterior-shape quantiles at that group's `max_disp_perc`.

Value

For `dGamma_list()`, an object of class "dGamma_list" (also inherits from "list"): a named list of "pfamily" objects (one per group). Attributes such as `window_diagnostics` and `measurement_prior_group` are retained for downstream use but are not shown by `print.dGamma_list()`. For `print.dGamma_list()`, `x` invisibly.

See Also

[dGamma](#), [Prior_Setup_GLMM](#), [pfamily_list](#), [rLMM_reg](#)

Examples

```
## dGamma_list() - per-group dGamma from Prior_Setup_GLMM (Ex_13b data)
##
## Requires dispformula = ~school_id so group.ing_prior is per-group.
## Requires suggested package bayesrules.

if (requireNamespace("bayesrules", quietly = TRUE)) {
  data(big_word_club, package = "bayesrules")

  dat <- big_word_club
  dat$school_id <- factor(dat$school_id)
  dat <- subset(
    dat,
    !is.na(score_ppvt) &
    !is.na(invalid_ppvt) & invalid_ppvt == 0L &
    complete.cases(dat[, c(
      "score_ppvt", "distracted_a1", "distracted_ppvt",
      "private_school", "title1", "free_reduced_lunch", "school_id"
    )])
  )

  form_lmer <- score_ppvt ~
    private_school + title1 + free_reduced_lunch +
    distracted_ppvt + distracted_a1 +
    free_reduced_lunch:distracted_a1 +
    (1 + distracted_ppvt + distracted_a1 || school_id)

  design0 <- model_setup(form_lmer, data = dat)
  dat <- subset(dat, school_id %in% names(design0$groupef.rank)[design0$groupef.rank])
  dat$school_id <- droplevels(dat$school_id)

  ps <- Prior_Setup_GLMM(
    form_lmer,
    data = dat,
    pop.pwt = 0.01,
    dispformula = ~school_id,
    group.max_disp_perc = 0.8,
    group.dispersion.pwt = 0.1,
    group.alpha_target = NULL
  )

  disp_pf <- dGamma_list(ps)

  ## Group names (school_id levels)
  g <- names(disp_pf)
  utils::head(g, 10L)

  ## Print a subset (recommended when J is large)
  print(disp_pf, groups = c("2", "18", "47"))

  ## Variation: rebuild truncation windows; print one group
  disp_pf_tight <- dGamma_list(ps, max_disp_perc_measurement = 0.9)
```

```

print(disp_pf_tight, groups = "2")

## Downstream attributes (not shown by print.dGamma_list)
names(attributes(disp_pf))
utils::head(attr(disp_pf, "window_diagnostics"), 3L)
}

```

directional_tail	<i>Directional Tail Diagnostic</i>
------------------	------------------------------------

Description

Computes the directional tail probability based on posterior draws and prior mean, using whitening transformation and projection onto the direction of disagreement. This diagnostic identifies directional disagreement between posterior and prior, and is especially useful for visualizing rejection regions in whitened space. The whitening uses Mahalanobis distance (Mahalanobis 1936) in posterior-precision-scaled coordinates.

Usage

```
directional_tail(fit, mu0 = NULL)
```

Arguments

<code>fit</code>	A fitted model object of class 'glmb' or 'lmb'
<code>mu0</code>	An optional argument containing a reference vector relative to which the directional tail is computed. Defaults to the prior mean.

Value

An object of class 'directional_tail' containing:

<code>mahalanobis_shift</code>	Measures the standardized Mahalanobis distance between the posterior and prior means, using posterior precision for scaling. In the Gaussian case, this directly determines the directional tail probability via $\Phi(- w)$.
<code>p_directional</code>	Directional tail probability (proportion of draws in the direction of disagreement)
<code>delta</code>	Mean shift in whitened space
<code>draws</code>	List containing whitened draws, raw draws, and tail flags

References

Mahalanobis PC (1936). "On the generalized distance in statistics." *Proceedings of the National Institute of Sciences of India*, 2(1), 49–55.

Nygren K (2025). "Chapter A04: Directional Tail Diagnostics for Prior-Posterior Disagreement." Vignette in the glmbayes R package. R vignette name: Chapter-A04.

See Also

[summary.glmb](#), [anova.glmb](#)

fixef.lmerb

Extract fixed effects from lmerb / glmerb fits

Description

By default returns [fixef](#) from the reference lmer/glmer fit (population-level regression coefficients). Use `type = "hyper"` or `"hyper.mean"` for Block~2 hyperparameter summaries stored on the Bayesian fit.

Usage

```
## S3 method for class 'lmerb'
fixef(object, type = c("mer", "hyper", "hyper.mean"), ...)

## S3 method for class 'glmerb'
fixef(object, type = c("mer", "hyper", "hyper.mean"), ...)
```

Arguments

<code>object</code>	An lmerb or glmerb object.
<code>type</code>	"mer" (default), "hyper" (popef.mode), or "hyper.mean" (popef.means; requires stored draws).
<code>...</code>	Ignored.

Value

A named numeric vector.

See Also

[fixef](#), [coef.lmerb](#).

glmb

*Fitting Bayesian Generalized Linear Models***Description**

glmb is used to fit Bayesian generalized linear models, specified by giving a symbolic descriptions of the linear predictor, the error distribution, and the prior distribution.

Usage

```
glmb(
  formula,
  family = binomial,
  pfamily = dNormal(mu, Sigma, dispersion = 1),
  n = 1000,
  data,
  weights,
  use_parallel = TRUE,
  use_opencl = FALSE,
  verbose = FALSE,
  subset,
  offset,
  na.action,
  Gridtype = 2,
  n_envopt = NULL,
  start = NULL,
  etastart,
  mustart,
  control = list(...),
  model = TRUE,
  method = "glm.fit",
  x = FALSE,
  y = TRUE,
  contrasts = NULL,
  ...
)
```

Arguments

formula	an object of class " formula " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	a description of the error distribution and link function to be used in the model. For <code>glm</code> this can be a character string naming a family function, a family function or the result of a call to a family function. For <code>glm.fit</code> only the third option is supported. (See family for details of family functions.)

<code>pfamily</code>	a description of the prior distribution and associated constants to be used in the model. This should be a <code>pfamily</code> function (see pfamily for details of <code>pfamily</code> functions).
<code>n</code>	number of draws to generate. If <code>length(n) > 1</code> , the length is taken to be the number required.
<code>data</code>	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>glm</code> is called.
<code>weights</code>	an optional vector of ‘prior weights’ to be used in the fitting process. Should be <code>NULL</code> or a numeric vector.
<code>use_parallel</code>	Logical. Whether to use parallel processing during simulation.
<code>use_opengl</code>	Logical. Whether to use OpenCL acceleration during Envelope construction.
<code>verbose</code>	Logical. Whether to print progress messages.
<code>subset</code>	an optional vector specifying a subset of observations to be used in the fitting process.
<code>offset</code>	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See documentation for <code>model.offset</code> at model.extract .
<code>na.action</code>	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of options , and is <code>na.fail</code> if that is unset. The ‘factory-fresh’ default is <code>stats{na.omit}</code> . Another possible value is <code>NULL</code> , no action. Value <code>stats{na.exclude}</code> can be useful.
<code>Gridtype</code>	an optional argument specifying the method used to determine the number of tangent points used to construct the enveloping function.
<code>n_envopt</code>	Effective sample size passed to <code>EnvelopeOpt</code> for grid construction. Defaults to match <code>n</code> . Larger values encourage tighter envelopes.
<code>start</code>	starting values for the parameters in the linear predictor.
<code>etastart</code>	starting values for the linear predictor.
<code>mustart</code>	starting values for the vector of means.
<code>control</code>	a list of parameters for controlling the fitting process. For <code>glm.fit</code> this is passed to glm.control .
<code>model</code>	a logical value indicating whether <i>model frame</i> should be included as a component of the returned value.
<code>method</code>	the method to be used in fitting the model. The default method <code>"glm.fit"</code> uses iteratively reweighted least squares (IWLS); the alternative <code>"model.frame"</code> returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as <code>glm.fit</code> . If specified as a character string it is looked up from within the stats namespace.

<code>x, y</code>	For <code>glm</code> : logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For <code>glm.fit</code> : <code>x</code> is a design matrix of dimension $n \times p$, and <code>y</code> is a vector of observations of length <code>n</code> .
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>...</code>	For <code>glm</code> : arguments to be used to form the default <code>control</code> argument if it is not supplied directly. For <code>weights</code> : further arguments passed to or from other methods.

Details

The function `glmb` is a Bayesian version of the classical `glm` function. The original R implementation of `glm` was written by Simon Davies (under Ross Ihaka at the University of Auckland) and has since been extensively rewritten by members of the R Core Team; its design was inspired by the `S` function described in (Hastie and Pregibon 1992), which in turn relies on the formula framework described in (Wilkinson and Rogers 1973).

Setup (including the use of formulas and families) mirrors that of `glm` but adds a required `pfamily` argument to specify the prior distribution. The design of the `pfamily` family of functions was created by Kjell Nygren and is modeled on how `glm` uses `family` to specify the likelihood.

For any implemented combination of family, link, and `pfamily`, `glmb` generates independent draws from the posterior density- no MCMC chains are required. Results can be printed or summarized with methods that mirror those for `glm` (e.g. `print.glmb`, `summary.glmb`), as well as all the usual `glm/lm` generics (`predict`, `residuals`, etc.).

A helper, `Prior_Setup`, assists users in choosing prior parameters. It ships with sensible defaults but also allows full customization. In particular, the default for `dNormal` is a reparameterization of Zellner's g-prior (Zellner 1986).

Currently supported response families are gaussian (identity link), poisson and quasipoisson (log link), gamma (log link), and binomial and quasibinomial (logit, probit, cloglog). All families support a `dNormal` prior; the Gaussian family also offers `dNormalGamma` and `dIndependent_Normal_Gamma`. Two conjugate `pfamilies` add closed-form IID sampling for intercept-only models with an identity link: `dBeta` for binomial(link = "identity") (Beta-Binomial conjugacy) and `dGamma`(`Inv_Dispersion` = FALSE) for poisson(link = "identity") and Gamma(link = "identity") (Gamma-Poisson and Gamma-Gamma rate conjugacy). For dispersion estimation with fixed coefficients, `dGamma` (default `Inv_Dispersion` = TRUE) places a Gamma prior on the inverse dispersion for Gaussian and Gamma(log) models.

For the Gaussian family, draws under `dNormal` and `dNormalGamma` come from posterior distributions resulting from conjugate prior distributions (Raiffa and Schlaifer 1961). For all other priors or response families, we use an accept-reject sampler built on the likelihood-subgradient envelope method (Nygren and Nygren 2006). The `Gridtype` argument controls how many tangent points are used in the envelope-trading off envelope tightness against construction cost-and `iters` reports candidate counts before acceptance.

By default, `glmb` draws $n = 1000$ samples, uses parallel CPU simulation, and-if `use_omp` = TRUE-GPU-accelerated envelope building. "`glmb`" comes with many of the same kinds of method functions that come with "`glm`" and "`lm`", so you can still call `extractAIC`, `fitted.values`, or any other standard method.

The `lmb` function is a Bayesian version of the `lm` function that can be used to estimate models from the Gaussian family without the need for a family argument.

`rglmb` and `rlmb` are functions with more minimalistic interfaces for estimating the same models without most of the internal overhead (these functions are called internally by `glmb` and `lmb`). The reduced overhead may be beneficial for Gibbs sampling implementations.

Value

`glmb` returns an object of class "glmb". The function `summary` (i.e., `summary.glmb`) can be used to obtain or print a summary of the results. The generic accessor functions `coefficients`, `fitted.values`, `residuals`, and `extractAIC` can be used to extract various useful features of the value returned by `glmb`.

An object of class "glmb" is a list containing at least the following components:

<code>glm</code>	an object of class "glm" containing the output from a call to the function <code>glm</code>
<code>coefficients</code>	a matrix of dimension <code>n</code> by <code>length(mu)</code> with one sample in each row
<code>coef.means</code>	a vector of <code>length(mu)</code> with the estimated posterior mean coefficients
<code>coef.mode</code>	a vector of <code>length(mu)</code> with the estimated posterior mode coefficients
<code>dispersion</code>	Either a constant provided as part of the call, or a vector of length <code>n</code> with one sample in each row.
<code>Prior</code>	A list with the priors specified for the model in question. Items in list may vary based on the type of prior
<code>fitted.values</code>	a matrix of dimension <code>n</code> by <code>length(y)</code> with one sample for the mean fitted values in each row
<code>family</code>	the <code>family</code> object used.
<code>linear.predictors</code>	an <code>n</code> by <code>length(y)</code> matrix with one sample for the linear fit on the link scale in each row
<code>deviance</code>	an <code>n</code> by 1 matrix with one sample for the deviance in each row
<code>pD</code>	An Estimate for the effective number of parameters
<code>Dbar</code>	Expected value for minus twice the log-likelihood function
<code>Dthetabar</code>	Value of minus twice the log-likelihood function evaluated at the mean value for the coefficients
<code>DIC</code>	Estimated Deviance Information criterion
<code>prior.weights</code>	a vector of weights specified or implied by the model
<code>y</code>	a vector with the dependent variable
<code>x</code>	a matrix with the implied design matrix for the model
<code>model</code>	if requested (the default), the model frame
<code>call</code>	the matched call
<code>formula</code>	the formula supplied
<code>terms</code>	the <code>terms</code> object used
<code>data</code>	the data argument

famfunc	Family functions used during estimation process
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting
digits	the number of significant digits to use when printing.

In addition, non-empty fits will have (yet to be implemented) components `qr`, `R` and effects relating to the final weighted linear fit for the posterior mode. Objects of class "glmb" are normal of class `c("glmb", "glm", "lm")`, that is inherit from classes `glm` and `lm` and well-designed methods from those classed will be applied when appropriate.

If a [binomial](#) glmb model was specified by giving a two-column response, the weights returned by `prior.weights` are the total number of cases (factored by the supplied case weights) and the component of `y` of the result is the proportion of successes.

References

Hastie T~J, Pregibon D (1992). "Generalized Linear Models." In Chambers J~M, Hastie T~J (eds.), *Statistical Models in S*, chapter 6. Wadsworth & Brooks/Cole, Belmont, CA.

Nygren K~N, Nygren L~M (2006). "Likelihood Subgradient Densities." *Journal of the American Statistical Association*, **101**(475), 1144–1156. doi:[10.1198/016214506000000357](#).

Raiffa H, Schlaifer R (1961). *Applied Statistical Decision Theory*. Clinton Press, Inc., Boston.

Wilkinson GN, Rogers CE (1973). "Symbolic Descriptions of Factorial Models for Analysis of Variance." *Applied Statistics*, **22**(3), 392–399. doi:[10.2307/2346786](#).

Zellner A (1986). "On Assessing Prior Distributions and Bayesian Regression Analysis with g-Prior Distributions." In Goel P~K, Zellner A (eds.), *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti*, volume 6 of *Studies in Bayesian Econometrics and Statistics*, 233–243. Elsevier. Dobson A~J (1990). *An Introduction to Generalized Linear Models*. Chapman and Hall, London.

See Also

[lm](#), [glm](#), [family](#), [formula](#) for classical modeling functions, family objects, and formula syntax

[pfamily](#) for documentation of pfamily functions used to specify priors.

[Prior_Setup](#), [Prior_Check](#) for functions used to initialize and to check priors,

[EnvelopeBuild](#) for envelope construction methods.

Further reading: (Nygren and Nygren 2006); (Nygren 2025, 2025); OpenCL/GPU: (Nygren 2025, 2025).

[summary.glmb](#), [predict.glmb](#), [residuals.glmb](#), [simulate.glmb](#), [extractAIC.glmb](#), [dummy.coef.glmb](#) and `methods(class="glmb")` for glmb and the methods and generic functions for classes `glm` and `lm` from which class `glmb` inherits.

glmbayes Modeling Functions [lmb\(\)](#)

Examples

```

set.seed(333)
## Dobson (1990) Page 93: Randomized Controlled Trial :
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
print(d.AD <- data.frame(treatment, outcome, counts))

## Classical Model
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson(link=log))
summary(glm.D93)

## Poisson Prior and Model
ps=Prior_Setup(counts ~ outcome + treatment,family = poisson())
mu=ps$mu
V=ps$Sigma

# Step 2: Call the glmb function
glmb.D93<-glmb(counts ~ outcome + treatment, family=poisson(),
               pfamily=dNormal(mu=mu,Sigma=V))
summary(glmb.D93)

# Menarche Binomial Data Example
data(menarche,package="MASS")
Age2=menarche$Age-13

## Classical Model

glm.out<-glm(cbind(Menarche, Total-Menarche) ~ Age2, family=binomial(logit), data=menarche)
summary(glm.out)

## Logit Prior and Model
ps1=Prior_Setup(cbind(Menarche, Total-Menarche) ~ Age2,family=binomial(logit), data=menarche)

glmb.out1<-glmb(cbind(Menarche, Total-Menarche) ~ Age2,
family=binomial(logit),pfamily=dNormal(mu=ps1$mu,Sigma=ps1$Sigma), data=menarche)
summary(glmb.out1)

## Posterior mean fitted probabilities on the response scale (see also
## \code{vignette("Chapter-05", package = "glmbayes")})
require(graphics)
pred1 <- predict(glmb.out1, type = "response")
pred1_m <- colMeans(pred1)
plot(
  Menarche / Total ~ Age,
  data = menarche,
  main = "Proportion with menarche (data and posterior mean fit)"
)
lines(menarche$Age, pred1_m, col = "blue", lwd = 2)

## Probit Prior and Model

```

```

ps2=Prior_Setup(cbind(Menarche, Total-Menarche) ~ Age2,family=binomial(probit), data=menarche)

glmb.out2<-glmb(cbind(Menarche, Total-Menarche) ~ Age2,
                family=binomial(probit),pfamily=dNormal(mu=ps2$mu,Sigma=ps2$Sigma), data=menarche)
summary(glmb.out2)

## clog-log Prior and Model
ps3=Prior_Setup(cbind(Menarche, Total-Menarche) ~ Age2,family=binomial(cloglog), data=menarche)

glmb.out3<-glmb(cbind(Menarche, Total-Menarche) ~ Age2,
                family=binomial(cloglog),pfamily=dNormal(mu=ps3$mu,Sigma=ps3$Sigma), data=menarche)
summary(glmb.out3)

## Comparison of DIC Statistics

DIC_Out=rbind(extractAIC(glmb.out1),extractAIC(glmb.out2),extractAIC(glmb.out3))
rownames(DIC_Out)=c("logit","probit","clog-log")
colnames(DIC_Out)=c("pD","DIC")
DIC_Out

### Gamma regression

data(carinsca)
carinsca$Merit <- ordered(carinsca$Merit)
carinsca$Class <- factor(carinsca$Class)
oldopt <- options(contrasts = c("contr.treatment", "contr.treatment"))
Claims=carinsca$Claims
Insured=carinsca$Insured
Merit=carinsca$Merit
Class=carinsca$Class
Cost=carinsca$Cost

out <- glm(Cost/Claims~Merit+Class,family=Gamma(link="log"),weights=Claims,x=TRUE)
summary(out)
disp=gamma.dispersion(out)
ps=Prior_Setup(Cost/Claims~Merit+Class,family=Gamma(link="log"),weights=Claims)
mu=ps$mu
V=ps$Sigma

out3 <- glmb(Cost/Claims~Merit+Class,family=Gamma(link="log"),
             pfamily=dNormal(mu=mu,Sigma=V,dispersion=disp),weights=Claims)

summary(out)
summary(out3)

options(oldopt)

## glmb with dGamma prior (dispersion-only; coefficients fixed)
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))

```

```

weight <- c(ctl, trt)
ps_dg <- Prior_Setup(weight ~ group, family = gaussian())
rate_dg <- if (!is.null(ps_dg$rate_gamma)) ps_dg$rate_gamma else ps_dg$rate
out_glmb_dGamma <- glmb(weight ~ group, family = gaussian(),
  pfamily = dGamma(shape = ps_dg$shape, rate = rate_dg, beta = ps_dg$coefficients))
summary(out_glmb_dGamma)

```

glmbBlock

Fitting Blocked Bayesian Generalized Linear Models

Description

Fits one [glmb](#) per observation block. Same row partition as [lmbBlock](#), but supports GLM [family](#) objects. Counterpart to [lmbBlock](#); see [summary.bglmb](#) for the summary method and [rNormalGLM_reg_group](#) for Gibbs sampling.

Usage

```

glmbBlock(
  formula,
  block,
  family = gaussian(),
  pfamily = NULL,
  pfamily_list = NULL,
  n = 1000,
  data,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  model = TRUE,
  x = FALSE,
  y = TRUE,
  Gridtype = 2,
  n_envopt = NULL,
  use_parallel = TRUE,
  use_opencil = FALSE,
  verbose = FALSE,
  ...
)

## S3 method for class 'bglmb'
print(x, digits = max(3, getOption("digits") - 3), ...)

```

Arguments

formula	an object of class " formula " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
block	Block partition: factor or vector of length <code>nrow(data)</code> (after <code>model.frame()</code>), a column name in data, <code>l2_blocks</code> counts, or a list of row index vectors (see normalize_group).
family	a description of the error distribution and link function to be used in the model. For <code>glm</code> this can be a character string naming a family function, a family function or the result of a call to a family function. For <code>glm.fit</code> only the third option is supported. (See family for details of family functions.)
pfamily	Recycled to all blocks, or use <code>pfamily_list</code> of length <code>k</code> .
pfamily_list	Optional list of <code>pfamily</code> objects, one per block.
n	number of draws to generate. If <code>length(n) > 1</code> , the length is taken to be the number required.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. If non- <code>NULL</code> , weighted least squares is used with weights <code>weights</code> (that is, minimizing $\sum(w \cdot e^2)$); otherwise ordinary least squares is used. See also 'Details'.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of options , and is <code>na.fail</code> if that is unset. The 'factory-fresh' default is <code>stats{na.omit}</code> . Another possible value is <code>NULL</code> , no action. Value <code>stats{na.exclude}</code> can be useful.
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector or matrix of extents matching those of the response. One or more offset terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See <code>model.offset</code> .
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
model, x, y	For <code>glmbBlock</code> , logicals passed to each block glmb fit (see glm). For <code>print</code> , <code>x</code> is the " <code>bglmb</code> " object.
Gridtype	an optional argument specifying the method used to determine the number of tangent points used to construct the enveloping function.
n_envopt	Effective sample size passed to <code>EnvelopeOpt</code> for grid construction. Defaults to match <code>n</code> . Larger values encourage tighter envelopes.
use_parallel	Logical. Whether to use parallel processing during simulation.
use_opengl	Logical. Whether to use OpenCL acceleration during Envelope construction.

verbose	Logical. Whether to print progress messages.
...	For <code>lm()</code> : additional arguments to be passed to the low level regression fitting functions (see below).
digits	Number of significant digits to use when printing.

Value

A named list of class "bglmb" (list of "glmb" fits from **glmbayes**).

Functions

- `glmbBlock()`: [glmb](#) fit per row block.

See Also

glmbayes Modeling Functions [lmbBlock\(\)](#)

glmerb	<i>Fit a Bayesian generalized linear mixed-effects model (GLMM) to data, via two-Block Gibbs sampling</i>
--------	---

Description

Bayesian generalized linear mixed-effects model fit

Usage

```
glmerb(
  formula,
  data = NULL,
  family = gaussian(),
  pfamily_list,
  dispersion_ranef = NULL,
  dispformula = ~1,
  n = 1000L,
  gap_tol = 0.0196,
  mode_gap_max = 1,
  tv_tol = 0.01,
  simulate = TRUE,
  REML = TRUE,
  control = NULL,
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
```

```

    contrasts = NULL,
    devFunOnly = FALSE,
    fixef = NULL,
    progbar = FALSE,
    sim_method = "DEFAULT",
    ...
)

## S3 method for class 'glmerb'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  sweep_history = FALSE,
  sweep_history_stage = c("main", "pilot", "both"),
  max_sweeps = Inf,
  ...
)

```

Arguments

formula	Mixed-model formula (single grouping factor; same constraints as model_setup).
data	Data frame containing all variables in formula.
family	A family object describing the response distribution and link. Defaults to <code>gaussian()</code> .
pfamily_list	Required named list of pfamily objects, one per random-effect coefficient (names must match the random-effect coefficient names, any order). Supplies the Block~2 hyperpriors (μ , Σ) and the Block~1 random-effect variances τ_k^2 . <code>dNormal</code> components treat τ_k^2 (the <code>pfamily</code> dispersion) as known and make conjugate γ_k draws. <code>dIndependent_Normal_Gamma</code> components place a Gamma prior on the Block~2 precision $1/\tau_k^2$: Block~2 then makes a joint (γ_k, τ_k^2) draw via the likelihood-subgradient envelope sampler (rinddepNormalGamma_reg), and the sampled τ_k^2 feeds back into the Block~1 prior precision. ING components must supply both truncation bounds: each τ_k^2 draw is hard-truncated to $[\text{disp_lower}, \text{disp_upper}]$, fixed across all inner Gibbs sweeps, with <code>disp_lower</code> doubling as the conservative τ_k^2 plug-in for the eigenvalue / TV calibration (smaller τ^2 increases the contraction rate λ^* , so the bound holds for every dispersion in the truncated support). They must also satisfy the prior-vs-data guard $n_{\text{prior}} \leq J$ (<code>pwt_dispersion</code> ≤ 0.5). Typically built with pfamily_list from a Prior_Setup_GLMM object.
dispersion_ranef	Observation-level measurement dispersion, treated as known during sampling. For families with a dispersion parameter (e.g. <code>gaussian()</code>): a positive scalar, a single pooled <code>dGamma()</code> , or (with <code>dispformula = ~<group_name></code>) a <code>dGamma_list(...)</code> or a named numeric vector of positive, fixed per-group values (names must match the random-effects grouping factor's levels exactly). Must be <code>NULL</code> (default) for <code>poisson()</code> and <code>binomial()</code> . Typically <code>Prior_Setup_GLMM(...)\$dispersion_ranef</code> . Which shapes are accepted depends on <code>dispformula</code> (see below). The fixed numeric vector shape currently only takes effect for <code>family = gaussian()</code> (shared

	code path with <code>lmerb</code>); observation-level dispersion is not yet wired into the sampler for the other dispersion families (<code>Gamma()</code> , <code>quasipoisson()</code> , <code>quasibinomial()</code>).
<code>dispformula</code>	One-sided formula selecting the measurement-dispersion structure: <code>~1</code> (default, pooled) requires <code>dispersion_ranef</code> to be a fixed scalar, <code>NULL</code> , or a single (pooled) <code>dGamma()</code> ; <code>~<group_name></code> , matching the random-effects grouping factor exactly, requires <code>dispersion_ranef</code> to be a <code>dGamma_list(...)</code> or a named numeric vector, and errors for families without a dispersion parameter. <code>~1</code> never fits an extra reference model. <code>~<group_name></code> with a <code>dGamma_list(...)</code> additionally requires a <code>glmmTMB</code> reference fit (glmmTMB must be installed), stored as <code>dispersion_fit</code> (reused from <code>dispersion_ranef</code> 's "dispersion_fit" attribute when <code>dispersion_ranef</code> was built via <code>dGamma_list(Prior_Setup_GLMM(..., dispformula = dispformula))</code> , rather than re-fitting <code>glmmTMB</code>); <code>~<group_name></code> with a fixed numeric vector never fits one, since the per-group dispersion is directly user-supplied, not a prior to calibrate. <code>glmer</code> is always the plain <code>glmer</code> fit regardless of <code>dispformula</code> .
<code>n</code>	Number of iid draws per group (default 1000L, as in <code>lmb</code>).
<code>gap_tol</code>	Legacy mode-mean gap tolerance. When <code>tv_tol</code> is <code>NULL</code> , the number of pilot chains is derived as <code>ceiling((qnorm(0.975) / gap_tol)^2)</code> (default <code>gap_tol = 0.0196</code> gives <code>n_pilot = 10000</code>). When <code>tv_tol</code> is set (default), <code>n_pilot</code> is instead chosen by <code>two_block_optimize_pilot_cost</code> to minimize total inner-sweep cost. Set <code>NULL</code> to skip the pilot unless <code>tv_tol</code> is set. Ignored for <code>gaussian()</code> without ING Block~2 components.
<code>mode_gap_max</code>	Maximum per-coordinate mode-mean gap (in posterior standard deviation units) used to calibrate pilot inner sweeps (default 1.0). Applies only to non-Gaussian families when <code>gap_tol</code> is not <code>NULL</code> . The pilot chains start at the ICM mode, which is at Mahalanobis distance $D_{\max} = \sqrt{p} \times \text{mode_gap_max}$ from the posterior mean (assuming <code>mode_gap_max</code> SDs per coordinate across p fixed-effect dimensions). The number of pilot sweeps is the smallest l satisfying $\text{erf}_1(0.5 \lambda^{*l} D_{\max} / \sqrt{2}) \leq \text{tv_tol}$ (Nygren 2020, Theorem 3 mean-shift term), floored at <code>m_min</code> . Set <code>mode_gap_max = NULL</code> to fall back to the Theorem~3 minimum sweep count. Ignored for <code>family = gaussian()</code> without ING Block~2 components.
<code>tv_tol</code>	Total variation tolerance per stored draw, in (0, 1) (default 0.01). For <code>family = gaussian()</code> the joint posterior is exactly multivariate normal and the number of inner Gibbs sweeps per stored draw is calibrated exactly as in <code>lmerb</code> (Nygren 2020, Theorem 3). For non-Gaussian families the same calibration is applied to the <i>local-Gaussian approximation of the posterior at its mode</i> : per-observation likelihood precisions are evaluated at the ICM posterior mode (<code>two_block_mode_weights()</code> in lmebayesCore) and fed to <code>two_block_rate_from_pfamily_list</code> . The derived sweep count is then the <i>minimum</i> number of iterations required to converge to that hypothetical multivariate normal approximation – a lower bound for the true (non-normal) posterior, not a guarantee.
<code>simulate</code>	Logical (default <code>TRUE</code>). Passed through to <code>rlmerb</code> . When <code>TRUE</code> the two-block Gibbs sampler is run for <code>n</code> iterations and posterior draws are stored. When <code>FALSE</code> , <code>rlmerb</code> returns fixed-VC plug-in point estimates only (exact joint Gaussian posterior mean). Simulation-only fields (<code>groupef</code> , <code>popef.means</code> , <code>popef</code>) are <code>NULL</code> when <code>simulate = FALSE</code> .

REML	Logical; passed to <code>model_setup</code> .
control	Optional <code>glmerControl</code> settings passed to the reference <code>glmer</code> fit. Defaults to NULL (lme4 defaults). When <code>family = gaussian()</code> , lme4's <code>glmer</code> shortcut to <code>lmer</code> does not accept an explicit <code>glmerControl</code> ; leave <code>control = NULL</code> or pass <code>lmerControl</code> .
start	Optional starting values; passed to <code>model_setup</code> .
verbose	Verbosity flag; passed to <code>model_setup</code> .
subset	Optional subset; passed to <code>model_setup</code> .
weights	Optional weights; passed to <code>model_setup</code> .
na.action	Missing-data handler; passed to <code>model_setup</code> .
offset	Optional offset; passed to <code>model_setup</code> .
contrasts	Optional contrasts; passed to <code>model_setup</code> .
devFunOnly	If TRUE, return deviance function only; passed to <code>model_setup</code> .
fixef	Optional named list of hyper-parameter vectors (Block 2 state). When NULL (default), iter-0 means are taken from the <code>pfamily_list</code> prior means.
progbar	Logical. When TRUE, show text progress bars during pilot and main replicate sampling inside <code>rglmerb</code> . Default FALSE.
sim_method	Sampling engine: "DEFAULT" or "TWO_BLOCK_GIBBS". Only changes behavior for <code>family = gaussian()</code> with fixed <code>dispersion_ranef</code> (a scalar or a named per-group vector) and all <code>dNormal()</code> <code>pfamily_list</code> components (known variance components) – see <code>lmerb</code> 's <code>sim_method</code> for details. Every other <code>glmerb</code> model (non-Gaussian families, or any <code>dIndependent_Normal_Gamma</code> component, or a sampled variance component) only has the two-block Gibbs engine, so both values behave identically there.
...	Reserved for future use.
x	Object of class "glmerb".
digits	Number of significant digits to use when printing.
sweep_history	If TRUE, print stored Block~2 sweep history (see <code>\$sweep_history</code>) after the usual summary.
sweep_history_stage	Which stage to print when <code>sweep_history = TRUE</code> : "main", "pilot", or "both".
max_sweeps	Passed to <code>print()</code> on sweep-history objects; limits how many inner sweeps are shown.

Details

Draft entry point for **lmebayes** GLMM models with a `glmer`-like interface, analogous to `lmerb` for Gaussian responses and to `glmb` for fixed-effects GLMs.

It takes the same arguments as `lmerb` plus `family`. With `family = gaussian()` it fits the same model as `lmerb`, differing only in that the embedded reference fit comes from `glmer` rather than `lmer`.

The model being estimated. For group $j = 1, \dots, J$ with link g ,

$$y_j \mid \beta_j \sim \text{family}(g^{-1}(D_j \beta_j)), \quad \beta_j \mid \gamma, \Psi \sim N(\mathcal{W}_j \gamma, \Psi),$$

where γ collects the level-2 fixed effects, Ψ is diagonal with entries τ_k^2 (one per random-effect coefficient), and $\mathcal{W}_j$ is built by `model_setup` from the level-2 predictors. As in `lmerb`, only a single grouping factor and uncorrelated ($||$) random-effect terms are supported. Marginalizing recovers the usual `glmer` form with $X_j = D_j \mathcal{W}_j$, $Z_j = D_j$, and mean-zero random effects $b_j = \beta_j - \mathcal{W}_j \gamma$.

Supported families are `gaussian()`, `poisson()`, `binomial()`, and `Gamma()` with the standard links. What they have in common is a *log-concave* likelihood, which is the condition the envelope accept-reject sampler requires (Nygren and Nygren 2006).

Because the link is applied within groups, β_j and γ live on the link scale — log rates for `poisson()`, log-odds for `binomial()` — so any summary on the response scale must apply the inverse link to the draws rather than to their averages.

What the priors decide. `pfamily_list` carries one component per random-effect coefficient and declares whether the between-group variance τ_k^2 is known (`dNormal`) or estimated (`dIndependent_Normal_Gamma`, under a Gamma prior on $1/\tau_k^2$ truncated to `[disp_lower, disp_upper]`). `dispersion_ranef` is the residual variance and applies to `gaussian()` only; it must be `NULL` for `poisson()` and `binomial()`, which have no free dispersion parameter. Both are normally built from one `Prior_Setup_GLMM` object.

What the returned draws are. As in `lmerb`, `glmerb` runs `n` independent replicate chains and stores one draw from each, taken at that chain's final sweep. The draws are therefore mutually independent — an iid sample, not an autocorrelated trace — so ordinary Monte Carlo standard errors apply and there is nothing to thin or diagnose for autocorrelation.

The accompanying guarantee is weaker here than for a Gaussian response, and the difference matters. With `family = gaussian()` and fixed variances the posterior is exactly normal and the sweep count is certified. A Poisson or binomial posterior is not normal, so the same calibration is applied at local curvature near the posterior mode. The resulting `m_convergence` is a well-motivated sweep *budget* rather than a proof, and `tv_tol` should be read that way. Tightening it is cheap: cost grows only logarithmically in $1/\text{tv_tol}$.

The same fact — that the mode of a skewed posterior is not its mean — is why a short **pilot stage** runs first. Its cross-chain average gives the main chains a better starting point than the mode does, which shortens the main stage. Pilot draws are diagnostic only and are never part of the returned sample. `gap_tol` and `mode_gap_max` are legacy controls used when `tv_tol` is `NULL`.

Point estimates (simulate = FALSE). `simulate = FALSE` returns the joint posterior *mode* by iterated conditional modes (the exact Gaussian mean when `family = gaussian()`), at fixed variance-component plug-ins. No sweeps run and no RNG is consumed, so the result is deterministic — useful while iterating on model specification.

Value

Object of class "glmerb": same `fixef.*` structure as "lmerb", including `sigma2` and `sigma2.mean` for `family = gaussian()` (see `lmerb`), with additional `family`, `glmer` (reference `glmer` fit, always the plain pooled-dispersion fit regardless of `dispformula`), `dispformula` (as supplied), `dispersion_fit` (`NULL` unless `dispformula` requests per-group dispersion, in which case the diagnostic-only `glmmTMB` fit), `fixef.init` (main-chain start from pilot `colMeans` when a pilot runs; `NULL` when no pilot runs), `pilot_chisq` (Hotelling chi-squared test of pilot mean vs ICM mode), `gap_tol`, and `mode_gap_max`.

References

- Nygren, K. (2020). *On the total variation distance between multivariate normal densities with applications to two-block Gibbs samplers*. Unpublished manuscript.
- Nygren, K. N. and Nygren, L. M. (2006). Likelihood Subgradient Densities. *Journal of the American Statistical Association* **101**(475), 1144–1156.
- Jones, G. L. and Hobert, J. P. (2001). Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science* **16**, 312–334.

See Also

`lmerb`, `glmerb_posterior_mode`, `glmb`; `demo` for the full sampling workflow (`demo("Ex_14_glmerb_airbnb_small", package = "lmebayes")`).

Examples

```
## glmerb: neighborhood random effects on bayesrules::airbnb_small (Poisson)
##
## Fast man-page example: prior setup, glmer reference fit, and ICM posterior
## mode only (simulate = FALSE). No Gibbs draws – suitable for R CMD check.
##
## The full sampling workflow (pilot stage, main draws, centering diagnostics)
## is preserved as a demo:
##   demo("Ex_14_glmerb_airbnb_small", package = "lmebayes")

if (requireNamespace("bayesrules", quietly = TRUE)) {

  data(airbnb_small, package = "bayesrules")
  dat <- airbnb_small
  dat$rating_c <- dat$rating - mean(dat$rating, na.rm = TRUE)
  dat$walk_c <- dat$walk_score - mean(dat$walk_score, na.rm = TRUE)
  dat <- dat[complete.cases(dat[, c("reviews", "rating_c", "walk_c",
                                   "neighborhood")]), ]
  dat$neighborhood <- droplevels(factor(dat$neighborhood))

  form <- reviews ~ walk_c + rating_c + (1 + rating_c || neighborhood)

  ps <- Prior_Setup_GLMM(form, data = dat, family = poisson(), pop.pwt = 0.01)

  fit <- glmerb(
    form,
    data      = dat,
    family    = poisson(),
    pfamily_list = pfamily_list(ps),
    simulate   = FALSE
  )

  lmebayes::print_coef_means(fit)
  print(fit)
  lme4::fixef(fit$glmer)
}
```

has_opengl	<i>OpenCL compile-time status for lmebayes</i>
------------	--

Description

Returns whether **this lmebayes** installation was built with OpenCL support (link-time / USE_OPENCL). This is independent of [has_opengl\(\)](#) in **opencltools**, which probes the host runtime (ICD, drivers, headers).

For workstation diagnostics (drivers, PATH, ICD), use [diagnose_glmbyes\(\)](#) from **opencltools**. For kernel loading from inst/cl/, use [load_kernel_source\(\)](#) with the appropriate package argument once **opencltools** is installed.

Usage

```
has_opengl()
```

Value

Logical scalar.

See Also

opencltools, **glmbayes**

lmb	<i>Fitting Bayesian Linear Models</i>
-----	---------------------------------------

Description

lmb is used to fit Bayesian linear models, specified by giving a symbolic descriptions of the linear predictor and the prior distribution.

Usage

```
lmb(
  formula,
  pfamily,
  n = 1000,
  data,
  subset,
  weights,
  na.action,
  method = "qr",
  model = TRUE,
  x = TRUE,
```

```

y = TRUE,
qr = TRUE,
singular.ok = TRUE,
contrasts = NULL,
offset,
Gridtype = 2,
n_envopt = NULL,
use_parallel = TRUE,
use_openc1 = FALSE,
verbose = FALSE,
...
)

```

Arguments

formula	an object of class " formula " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
pfamily	a description of the prior distribution and associated constants to be used in the model. For a single-response formula this should be a single pfamily object. For a multi-response formula (e.g. <code>cbind(y1, y2) ~ x</code>) this must be a list of pfamily objects with exactly one entry per response column; passing a single pfamily object is an error.
n	number of draws to generate. If <code>length(n) > 1</code> , the length is taken to be the number required.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. If non- <code>NULL</code> , weighted least squares is used with weights <code>weights</code> (that is, minimizing $\sum(w \cdot e^2)$); otherwise ordinary least squares is used. See also 'Details',
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of options , and is na.fail if that is unset. The 'factory-fresh' default is <code>stats{na.omit}</code> . Another possible value is <code>NULL</code> , no action. Value <code>stats{na.exclude}</code> can be useful.
method	the method to be used in fitting the classical model during a call to glm . The default method <code>glm.fit</code> uses iteratively reweighted least squares (IWLS); the alternative " <code>model.frame</code> " returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as <code>glm.fit</code> . If specified as a character string it is looked up from within the stats namespace.

<code>model, x, y, qr</code>	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
<code>singular.ok</code>	logical. If FALSE (the default in S but not in R) a singular fit is an error.
<code>contrasts</code>	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
<code>offset</code>	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector or matrix of extents matching those of the response. One or more offset terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See <code>model.offset</code> .
<code>Gridtype</code>	an optional argument specifying the method used to determine the number of tangent points used to construct the enveloping function.
<code>n_envopt</code>	Effective sample size passed to <code>EnvelopeOpt</code> for grid construction. Defaults to match <code>n</code> . Larger values encourage tighter envelopes.
<code>use_parallel</code>	Logical. Whether to use parallel processing during simulation.
<code>use_opengl</code>	Logical. Whether to use OpenCL acceleration during Envelope construction.
<code>verbose</code>	Logical. Whether to print progress messages.
<code>...</code>	For <code>lm()</code> : additional arguments to be passed to the low level regression fitting functions (see below).

Details

The function `lmb` is a Bayesian extension of the classical `lm` function. It retains the familiar formula interface and model setup used in `lm`, while introducing posterior simulation and prior specification via the `pfamily` argument. Internally, `lmb` calls `lm` to obtain the classical least squares fit, then generates independent draws from the posterior distribution using either multivariate normal simulation (for Gaussian priors) or accept-reject sampling via likelihood-subgradient envelopes (Nygren and Nygren 2006).

The symbolic formula interface follows (Wilkinson and Rogers 1973), and the overall design of `lm` was inspired by the S system (Chambers 1992). `lmb` comes with many of the same types of generic methods that are available to `lm` and `glm`, including `predict`, `residuals`, `extractAIC`, and `summary`. Many of these are inherited from `glm`.

Prior specification is handled via the `pfamily` argument, which defines the prior mean, covariance, and dispersion. The design of the `pfamily` family of functions was created by Kjell Nygren and is modeled on how `glm` uses `family` to specify the likelihood. A helper function, `Prior_Setup`, assists users in choosing prior parameters. It ships with sensible defaults but also allows full customization. All models support the `dNormal` prior; the Gaussian family also supports `dNormalGamma` and `dIndependent_Normal_Gamma`, which allow for more flexible prior structures including independent priors on variance components.

Posterior draws are generated using the prior specification provided via `pfamily`. For Gaussian models with conjugate priors, draws are obtained directly from the posterior distribution using standard simulation procedures for multivariate normal densities (Raiffa and Schlaifer 1961). For non-conjugate setups, the function uses envelope-based accept-reject sampling, where the `Gridtype` parameter controls the granularity of the envelope construction. The number of candidates generated before acceptance is returned in the `iters` component.

The output includes both the classical `lm` fit and Bayesian diagnostics such as the Deviance Information Criterion (DIC), effective number of parameters (pD), and posterior summaries. The DIC, introduced by (Spiegelhalter et al. 2002), provides a Bayesian analog to AIC by balancing model fit and complexity using posterior expectations. This dual structure allows users to compare classical and Bayesian fits side-by-side, and to leverage familiar modeling workflows while gaining access to richer inferential tools.

The `lmb` function is a specialized version of `glmb` for Gaussian models, and does not require a family argument. For conjugate models, it uses standard simulation methods for posterior draws, avoiding the need for envelope construction or subgradient sampling. Like `glmb`, it returns objects compatible with many standard methods from `lm` and `glm`, including `extractAIC`, `fitted.values`, and `residuals`.

For more minimalistic workflows, `rlmb` and `rglmb` offer stripped-down interfaces for posterior sampling without the overhead of full model objects. `rlmb` is called from within `lmb`. The functions `rlmb` might be useful in Gibbs sampling or simulation-heavy contexts.

Value

`lmb` returns an object of class "lmb". The function `summary` (i.e., `summary.glmb`) can be used to obtain or print a summary of the results. The generic accessor functions `coefficients`, `fitted.values`, `residuals`, and `extractAIC` can be used to extract various useful features of the value returned by `lmb`.

An object of class "lmb" is a list containing at least the following components:

<code>lm</code>	an object of class "lm" containing the output from a call to the function <code>lm</code>
<code>coefficients</code>	a matrix of dimension <code>n</code> by <code>length(mu)</code> with one sample in each row
<code>coef.means</code>	a vector of <code>length(mu)</code> with the estimated posterior mean coefficients
<code>coef.mode</code>	a vector of <code>length(mu)</code> with the estimated posterior mode coefficients
<code>dispersion</code>	Either a constant provided as part of the call, or a vector of length <code>n</code> with one sample in each row.
<code>Prior</code>	A list with the priors specified for the model in question. Items in list may vary based on the type of prior
<code>residuals</code>	a matrix of dimension <code>n</code> by <code>length(y)</code> with one sample for the deviance residuals in each row
<code>fitted.values</code>	a matrix of dimension <code>n</code> by <code>length(y)</code> with one sample for the fitted values in each row
<code>linear.predictors</code>	an <code>n</code> by <code>length(y)</code> matrix with one sample for the linear fit on the link scale in each row
<code>deviance</code>	an <code>n</code> by 1 matrix with one sample for the deviance in each row
<code>pD</code>	An Estimate for the effective number of parameters
<code>Dbar</code>	Expected value for minus twice the log-likelihood function
<code>Dthetabar</code>	Value of minus twice the log-likelihood function evaluated at the mean value for the coefficients
<code>DIC</code>	Estimated Deviance Information criterion

weights	a vector of weights specified or implied by the model
prior.weights	a vector of weights specified or implied by the model
y	a vector of observations of length m.
x	a design matrix of dimension $m \times p$
model	if requested (the default), the model frame
call	the matched call
formula	the formula supplied
terms	the terms object used
data	the data argument
famfunc	family functions used during estimation and post processing
iters	an n by 1 matrix giving the number of candidates generated before acceptance for each sample.
contrasts	(where relevant) the contrasts used.
xlevels	(where relevant) a record of the levels of the factors used in fitting
pfamily	the prior family specified
digits	the number of significant digits to use when printing.

In addition, non-empty fits will have (yet to be implemented) components `qr`, `R` and `effects` relating to the final weighted linear fit for the posterior mode. Objects of class "lmb" are normally of class `c("lmb", "glmb", "glm", "lm")`, that is inherit from classes `glmb`, `glm` and `lm` and well-designed methods from those classes will be applied when appropriate.

References

- Chambers JM (1992). "Linear Models." In Chambers JM, Hastie TJ (eds.), *Statistical Models in S*, chapter 4, 85–124. Wadsworth & Brooks/Cole, Pacific Grove, CA.
- Nygren K~N, Nygren L~M (2006). "Likelihood Subgradient Densities." *Journal of the American Statistical Association*, **101**(475), 1144–1156. doi:[10.1198/016214506000000357](#).
- Raiffa H, Schlaifer R (1961). *Applied Statistical Decision Theory*. Clinton Press, Inc., Boston.
- Spiegelhalter DJ, Best NG, Carlin BP, van der Linde A (2002). "Bayesian Measures of Model Complexity and Fit." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **64**(4), 583–639. doi:[10.1111/14679868.00353](#).
- Wilkinson GN, Rogers CE (1973). "Symbolic Descriptions of Factorial Models for Analysis of Variance." *Applied Statistics*, **22**(3), 392–399. doi:[10.2307/2346786](#).

See Also

The classical modeling functions [lm](#) and [glm](#).
[glmb](#), [rglmb](#), [rlmb](#) for related Bayesian GLM/linear interfaces; [EnvelopeBuild](#) for envelope construction when accept–reject sampling is used.

[pfamily](#) for documentation of pfamily functions used to specify priors.

[Prior_Setup](#), [Prior_Check](#) for functions used to initialize and to check priors,

Further reading: (Nygren and Nygren 2006); (Nygren 2025, 2025); independent Normal–Gamma sampler: (Nygren 2025).

[summary.glmb](#), [predict.glmb](#), [simulate.glmb](#), [extractAIC.glmb](#), [dummy.coef.glmb](#) and `methods(class="glmb")` for methods inherited from class `glmb` and the methods and generic functions for classes `glm` and `lm` from which class `lmb` also inherits.

glmbayes Modeling Functions [glmb\(\)](#)

Examples

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)

ps <- Prior_Setup(weight ~ group, gaussian())
# Prior_Setup supplies the prior mean and covariance components used below.

## Classical model
lm.D9 <- lm(weight ~ group, x = TRUE, y = TRUE)
summary(lm.D9)
vcov(lm.D9)
s <- summary(lm.D9)
disp_classical <- s$sigma^2
cat("Classical lm dispersion (sigma^2 = RSS/(n-p)):", disp_classical, "\n")

## Conjugate Normal Prior (fixed dispersion)
lmb.D9 <- lmb(
  weight ~ group,
  pfamily = dNormal(mu = ps$mu, ps$Sigma, dispersion = ps$dispersion)
)
summary(lmb.D9)
vcov(lmb.D9)

## Conjugate Normal_Gamma Prior (second argument is Sigma_0 from Prior_Setup)
lmb.D9_v2 <- lmb(
  weight ~ group,
  pfamily = dNormal_Gamma(
    ps$mu,
    Sigma_0 = ps$Sigma_0,
    shape = ps$shape,
    rate = ps$rate
  )
)
summary(lmb.D9_v2)
vcov(lmb.D9_v2)

## Independent_Normal_Gamma_Prior (same mu, Sigma, rate as dNormal_Gamma; shape = ps$shape_ING)
```

```

lmb.D9_v3 <- lmb(
  weight ~ group,
  dIndependent_Normal_Gamma(
    ps$mu,
    ps$Sigma,
    shape = ps$shape_ING,
    rate = ps$rate
  )
)
summary(lmb.D9_v3)

## anova
anova(lmb.D9)

## lmb with dGamma prior (dispersion-only; coefficients fixed)
rate_dg <- if (!is.null(ps$rate_gamma)) ps$rate_gamma else ps$rate
out_lmb_dGamma <- lmb(
  weight ~ group,
  pfamily = dGamma(shape = ps$shape, rate = rate_dg, beta = ps$coefficients))
summary(out_lmb_dGamma)

```

lmbBlock

Fitting Blocked Bayesian Linear Models

Description

Fits one [lmb](#) per observation block (SAS BY-style split on rows), sharing the same formula on each subset. Contrast with [lmb](#) on a `cbind(...)` response (several response columns) and [rNormalGLM_reg_group](#) (Gibbs conditional draws, matrix API).

Usage

```

lmbBlock(
  formula,
  block,
  pfamily = NULL,
  pfamily_list = NULL,
  n = 1000,
  data,
  subset,
  weights,
  na.action,
  method = "qr",
  model = TRUE,
  x = TRUE,
  y = TRUE,
  qr = TRUE,

```

```

singular.ok = TRUE,
contrasts = NULL,
offset,
Gridtype = 2,
n_envopt = NULL,
use_parallel = TRUE,
use_openc1 = FALSE,
verbose = FALSE,
...
)

## S3 method for class 'lmb'
print(x, digits = max(3, getOption("digits") - 3), ...)

```

Arguments

formula	an object of class " formula " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
block	Block partition: factor or vector of length <code>nrow(data)</code> (after <code>model.frame</code>), a column name in data, <code>l2_blocks</code> counts, or a list of row index vectors (see normalize_group).
pfamily	A single pfamily recycled to every block, or use <code>pfamily_list</code> of length <code>k</code> (number of blocks).
pfamily_list	Optional list of <code>pfamily</code> objects, one per block.
n	number of draws to generate. If <code>length(n) > 1</code> , the length is taken to be the number required.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>lm</code> is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
weights	an optional vector of weights to be used in the fitting process. Should be <code>NULL</code> or a numeric vector. If non- <code>NULL</code> , weighted least squares is used with weights <code>weights</code> (that is, minimizing $\sum(w \cdot e^2)$); otherwise ordinary least squares is used. See also 'Details'.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the <code>na.action</code> setting of options , and is <code>na.fail</code> if that is unset. The 'factory-fresh' default is <code>stats{na.omit}</code> . Another possible value is <code>NULL</code> , no action. Value <code>stats{na.exclude}</code> can be useful.
method	the method to be used in fitting the classical model during a call to glm . The default method <code>glm.fit</code> uses iteratively reweighted least squares (IWLS); the alternative " <code>model.frame</code> " returns the model frame and does no fitting. User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as

	glm.fit. If specified as a character string it is looked up from within the stats namespace.
model, x, y, qr	For lmbBlock, logicals passed to each block lmb fit (see lm). For print, x is the "blmb" object.
singular.ok	logical. If FALSE (the default in S but not in R) a singular fit is an error.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector or matrix of extents matching those of the response. One or more offset terms can be included in the formula instead or as well, and if more than one are specified their sum is used. See model.offset.
Gridtype	an optional argument specifying the method used to determine the number of tangent points used to construct the enveloping function.
n_envopt	Effective sample size passed to EnvelopeOpt for grid construction. Defaults to match n. Larger values encourage tighter envelopes.
use_parallel	Logical. Whether to use parallel processing during simulation.
use_opengl	Logical. Whether to use OpenCL acceleration during Envelope construction.
verbose	Logical. Whether to print progress messages.
...	For lm(): additional arguments to be passed to the low level regression fitting functions (see below).
digits	Number of significant digits to use when printing.

Value

A named list of class "blmb" (list of "lmb" fits).

Functions

- lmbBlock(): Gaussian [lmb](#) fit per row block.

See Also

glmbayes Modeling Functions [glmbBlock\(\)](#)

Examples

```
## Row-block lmb (iris): BY Species - SAS-style separate regressions

set.seed(42)
data("iris", package = "datasets")

ps_block <- Prior_SetupGroup(
  Sepal.Length ~ Sepal.Width + Petal.Length,
  group = "Species",
  data = iris,
  family = gaussian()
)
```

```

out_blmb <- lmbBlock(
  Sepal.Length ~ Sepal.Width + Petal.Length,
  block = "Species",
  pfamily_list = pfamily_list(ps_block),
  data = iris,
  n = 150L,
  use_parallel = FALSE
)

print(out_blmb)
summary(out_blmb)

```

lmerb

Fit a Bayesian linear mixed-effects model (LMM) to data, via two-Block Gibbs sampling

Description

Bayesian linear mixed-effects model fit

Usage

```

lmerb(
  formula,
  data = NULL,
  pfamily_list,
  dispersion_ranef,
  dispformula = ~1,
  n = 1000L,
  tv_tol = 0.01,
  gap_tol = 0.0196,
  mode_gap_max = 1,
  diag_sweeps = FALSE,
  progbar = NULL,
  sim_method = "DEFAULT",
  simulate = TRUE,
  REML = TRUE,
  control = lme4::lmerControl(),
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  devFunOnly = FALSE,
  fixef = NULL,

```

```

    ...
)

## S3 method for class 'lmerb'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  sweep_history = FALSE,
  sweep_history_stage = c("main", "pilot", "both"),
  max_sweeps = Inf,
  components = NULL,
  covariate = NULL,
  ...
)

```

Arguments

formula	Mixed-model formula (single grouping factor; same constraints as model_setup).
data	Data frame containing all variables in formula.
pfamily_list	Required named list of pfamily objects, one per random-effect coefficient (names must match the random-effect coefficient names, any order). Supplies the Block~2 hyperpriors (μ , Σ) and the Block~1 random-effect variances τ_k^2 . dNormal components treat τ_k^2 (the pfamily dispersion) as known and make conjugate γ_k draws. dIndependent_Normal_Gamma components place a Gamma prior on the Block~2 precision $1/\tau_k^2$: Block~2 then makes a joint (γ_k, τ_k^2) draw via the likelihood-subgradient envelope sampler (rinddepNormalGamma_reg), and the sampled τ_k^2 feeds back into the Block~1 prior precision. ING components must supply both truncation bounds: each τ_k^2 draw is hard-truncated to $[\text{disp_lower}, \text{disp_upper}]$, fixed across all inner Gibbs sweeps, with disp_lower doubling as the conservative τ_k^2 plug-in for the eigenvalue / TV calibration (smaller τ^2 increases the contraction rate λ^* , so the bound holds for every dispersion in the truncated support). They must also satisfy the prior-vs-data guard $n_{\text{prior}} \leq J$ ($\text{pwt_dispersion} \leq 0.5$). Typically built with pfamily_list from a Prior_Setup_GLMM object.
dispersion_ranef	Observation-level measurement dispersion σ^2 for Block~1. One of: a positive scalar (treated as known; typically <code>Prior_Setup_GLMM(...)\$dispersion_ranef</code>), a single dGamma() pfamily (pooled σ^2 across groups), a named list of dGamma() objects (one per group level) from dGamma_list(Prior_Setup_GLMM(...)) , or a named numeric vector of positive, fixed per-group values (names must match the random-effects grouping factor's levels exactly; each group's σ_j^2 is then treated as known, like the pooled scalar case but allowed to vary by group). Which of these four shapes is accepted depends on <code>dispformula</code> (see below).
dispformula	One-sided formula selecting the measurement-dispersion structure: <code>~1</code> (default, pooled) requires <code>dispersion_ranef</code> to be a fixed scalar or a single (pooled) dGamma() ; <code>~<group_name></code> , matching the random-effects grouping factor exactly, requires <code>dispersion_ranef</code> to be a dGamma_list(...) (one dGamma() per group level) or a named numeric vector (one fixed value per group level).

Any other formula is an error. `~1` never fits an extra reference model. `~<group_name>` with a `dGamma_list(...)` additionally requires a `glmmTMB` reference fit (**glmmTMB** must be installed), stored as `dispersion_fit`; `~<group_name>` with a fixed numeric vector never fits one, since the per-group dispersion is directly user-supplied, not a prior to calibrate. When `dispersion_ranef = dGamma_list(Prior_Setup_GLMM(..., dispformula = dispformula))`, that call already fit the `glmmTMB` reference model to calibrate the priors, and `lmerb()` reuses it here rather than fitting `glmmTMB` a second time; keep `dispformula` identical between the two calls, since it is not re-validated against the reused fit. `lmer` is always the plain `lmer` fit regardless of `dispformula`; the sampler route (pooled vs. per-group) already follows from `dispersion_ranef`'s shape alone.

<code>n</code>	Number of iid draws per group (default 1000L, as in <code>lmb</code>).
<code>tv_tol</code>	Total variation tolerance per stored draw, in (0, 1) (default 0.01, the conventional threshold of the honest-burn-in literature; Jones and Hobert 2001). The number of inner Gibbs sweeps per stored draw is derived so that each draw is within <code>tv_tol</code> of the exact joint posterior in total variation (Nygren 2020, Theorem 3; see Details). To certify the whole <code>n</code> -draw sample at level α pass <code>tv_tol = alpha / n</code> ; the cost grows only logarithmically in $1/\text{tv_tol}$.
<code>gap_tol</code>	Legacy mode—mean gap tolerance for the pilot stage when any Block~2 component uses <code>dIndependent_Normal_Gamma</code> and <code>tv_tol</code> is NULL. When <code>tv_tol</code> is set (default), the pilot chain count is chosen by cost optimization instead. Set NULL to skip the pilot unless <code>tv_tol</code> is set. Ignored for all- <code>dNormal</code> models.
<code>mode_gap_max</code>	Maximum per-coordinate mode—mean gap (in posterior standard deviation units) used to calibrate pilot inner sweeps when ING Block~2 components run a pilot stage (default 1.0). Ignored for all- <code>dNormal</code> models.
<code>diag_sweeps</code>	Temporary diagnostic flag (ING models with pilot). Non- <code>dNormal</code> sampling already runs via <code>lmebayesCore::run_sweep_outer_chains_v6()</code> (R sweep-outer; pilot then main). When TRUE, each stage auto-prints one combined Block~2 chain-mean table when that stage finishes (same layout as <code>print()</code> on <code>\$sweep_history</code>). <code>sweep_history</code> is collected regardless. Default FALSE.
<code>progbar</code>	Logical. Show text progress bars during sampling (passed to <code>rlmerb</code>). Default NULL: FALSE when <code>diag_sweeps = TRUE</code> , otherwise TRUE.
<code>sim_method</code>	Sampling engine: "DEFAULT" or "TWO_BLOCK_GIBBS". Only changes behavior when <code>dispersion_ranef</code> is fixed (a scalar or a named per-group vector) and every <code>pfamily_list</code> component is <code>dNormal()</code> (known variance components) — the joint posterior is then exactly multivariate normal, and "DEFAULT" draws directly, iid, from that closed form (no Gibbs sweeps, no burn-in, no autocorrelation between draws); "TWO_BLOCK_GIBBS" forces the two-block Gibbs sampler described above instead. Every other model (any <code>dIndependent_Normal_Gamma</code> component, or a sampled/estimated variance component) only has the two-block Gibbs engine, so both values behave identically there. See <code>rlMMNormal_reg_known_vcov</code> .
<code>simulate</code>	Logical (default TRUE). Passed through to <code>rlmerb</code> . When TRUE the two-block Gibbs sampler is run for <code>n</code> iterations and posterior draws are stored. When FALSE, <code>rlmerb</code> returns fixed-VC plug-in point estimates only (exact joint Gaussian posterior mean). Simulation-only fields (<code>groupef</code> , <code>popef.means</code> , <code>popef</code>) are NULL when <code>simulate = FALSE</code> .

REML	Logical; passed to <code>model_setup</code> .
control	<code>lmerControl</code> settings; passed to <code>model_setup</code> .
start	Optional starting values; passed to <code>model_setup</code> .
verbose	Verbosity flag; passed to <code>model_setup</code> .
subset	Optional subset; passed to <code>model_setup</code> .
weights	Optional weights; passed to <code>model_setup</code> .
na.action	Missing-data handler; passed to <code>model_setup</code> .
offset	Optional offset; passed to <code>model_setup</code> .
contrasts	Optional contrasts; passed to <code>model_setup</code> .
devFunOnly	If TRUE, return deviance function only; passed to <code>model_setup</code> .
fixef	Optional named list of hyper-parameter vectors (Block 2 state). When NULL (default), iter-0 means are taken from the <code>pfamily_list</code> prior means.
...	Reserved for future use.
x	Object of class "lmerb".
digits	Number of significant digits to use when printing.
sweep_history	If TRUE, print stored Block~2 sweep history (see <code>\$sweep_history</code>) after the usual summary: one combined table per stage with mode plus all inner sweeps.
sweep_history_stage	Which stage to print when <code>sweep_history = TRUE</code> : "main", "pilot", or "both".
max_sweeps	Passed to <code>print()</code> on sweep-history objects; limits how many inner sweeps are shown per stage.
components	Optional RE components passed to sweep-history <code>print()</code> .
covariate	Optional covariate names passed to sweep-history <code>print()</code> .

Details

Fits a Bayesian linear mixed model from an `lmer`-style formula, returning posterior draws. It is the mixed-model counterpart of `lmb` for fixed-effects models.

`lmerb` builds the design with `model_setup`, embeds the companion `lmer` fit for reference, and passes both to the `lmebayesCore` sampler. Priors are supplied as `pfamily_list` (population level) and `dispersion_ranef` (residual variance), normally built from one `Prior_Setup_GLMM` object.

The model being estimated:

For group $j = 1, \dots, J$, `lmerb` estimates

$$y_j \mid \beta_j, \sigma^2 \sim N(D_j \beta_j, \sigma^2 I_{n_j}) \quad (\text{within group})$$

$$\beta_j \mid \gamma, \Psi \sim N(\mathcal{W}_j \gamma, \Psi) \quad (\text{across groups})$$

The second line is the model for how groups differ: each group's own coefficient vector β_j varies around a population expectation $\mathcal{W}_j \gamma$, and Ψ measures how widely. This is a *centered* parameterization — the full coefficient β_j enters the likelihood, not a mean-zero deviation — so `coef()` on the fit reports β_j directly and `ranef()` reports the deviation $\beta_j - \mathcal{W}_j \gamma$ that **lme4** calls b_j . Marginalizing recovers the usual **lme4** model, with γ playing the role of the fixed effects.

Ψ is diagonal because formula may only contain uncorrelated (`| |`) random-effect terms and a single grouping factor; there is no random-effect correlation to estimate. See `vignette("Chapter-02", package = "lmebayes")` for the formula rules.

What the priors decide:

The two prior arguments are where you declare which variances are known and which are estimated, and that choice determines what comes back.

`pfamily_list` One component per random-effect coefficient. A `dNormal` component fixes the between-group variance τ_k^2 at the supplied value; a `dIndependent_Normal_Gamma` component estimates it, under a Gamma prior on $1/\tau_k^2$ truncated to `[disp_lower, disp_upper]`.

`dispersion_ranef` The residual variance σ^2 : a fixed scalar, a pooled `dGamma()` prior to estimate one common value, a `dGamma_list()` to estimate a separate σ_j^2 per group, or a named vector of fixed per-group values. `dispformula` must agree with the shape supplied.

When every variance is fixed, the posterior is exactly multivariate normal and the draws are exact. Estimating any variance makes the posterior non-Gaussian and the draws approximate in the sense described next.

What the returned draws are:

`lmerb` does not return one long chain. It runs `n` **independent replicate chains** and keeps **one draw from each**, taken at that chain's final sweep. Two things follow:

1. **The draws are mutually independent.** They form an iid sample, so ordinary Monte Carlo standard errors apply. Nothing needs thinning, there is no effective sample size to compute, and trace plots or $\hat{R}$ on the returned draws are not meaningful.
2. **Each draw comes from a distribution deliberately made close to the posterior.** The sweep count is not a burn-in guess: it is calibrated before sampling so that the distribution reached at the final sweep is within `tv_tol` of the exact posterior in total variation. The resulting `m_convergence` is reported on the fit.

`n` and `tv_tol` therefore control different errors — Monte Carlo noise and per-draw bias — and can be set independently. To certify all `n` draws jointly at level α , pass `tv_tol = alpha / n`; cost grows only logarithmically in $1/\text{tv_tol}$, so this usually adds a few sweeps.

The calibration itself, and what to do if a warning reports slow convergence, are covered in `vignette("Chapter-06", package = "lmebayes")`.

Value

Object of class `"lmerb"`: a list with the following components (parallel to `glmb` and `lmb`).

Every draw slot holds one row per replicate chain, taken at that chain's final sweep, so the rows are mutually independent and can be averaged directly. Accessors `fixef()`, `ranef()`, `coef()` and `summary()` are usually more convenient than reaching into these fields; see `vignette("Chapter-05", package = "lmebayes")`.

`call` The matched call.

`formula` The formula supplied.

`lmer` `lmer` fit from `model_setup` (full formula), embedded as a sub-object — analogous to `glmb$glm` and `lmb$lm`. Use `coef(fit$lmer)` for per-group classical coefficients. Always the plain pooled-dispersion `lmer` fit, regardless of `dispformula`.

`dispformula` The `dispformula` supplied.

`dispersion_fit` NULL when `dispformula = ~1`; otherwise the diagnostic-only `glmmTMB` fit with per-group residual dispersion (`dispformula`). Identical to `glmmTMB` below; kept for backward compatibility.

`glmmTMB` Same value as `dispersion_fit` (an alias): `NULL` when `dispformula = ~1`, otherwise the `glmmTMB` per-group-dispersion reference fit. Prefer this field name in new code; downstream consumers that need a per-group-dispersion-aware reference fit (e.g. `summary.lmerb()`) use `glmmTMB` when non-`NULL` and fall back to `lmer` otherwise.

`prior` Normalized prior container: `pfamily_list` (as supplied, reordered to the RE coefficient names), `dispersion_ranef`, the reconstructed `Sigma_ranef`, and the per-component `prior_list` (`mu_fixef`, `Sigma_fixef`, `dispersion_fixef`) — analogous to `glmb$Prior`.

`model_setup` The `model_setup` object built inside `lmerb` from `formula` and `data`.

`popgef.mode` Named list of exact posterior mode (= mean, since the joint posterior is Gaussian) vectors for the level-2 fixed effects γ_k , computed by `lmerb_posterior_mean` (ICM).

`grougef.mode` $J \times p_{re}$ numeric matrix of exact posterior mode random effects from ICM. Rows are group levels (`levels(design$group)`); columns are `design$re_coef_names`.

`popgef` Named list of $n \times q_k$ matrices of population draws γ_k , one per random-effect coefficient. `NULL` when `simulate = FALSE`.

`popgef.means` Named list of posterior mean vectors, the column means of `popgef`. `NULL` when `simulate = FALSE`.

`grougef` `data.frame` with $n \times J$ rows: draw, the grouping-factor column, and one column per RE variable. These are full group coefficients β_j , comparable to `lme4::coef()`, not mean-zero deviations. Average over draw within each group for posterior means (see Examples). `NULL` when `simulate = FALSE`.

`popgef.dispersion` $n \times p_{re}$ matrix of the between-group variances τ_k^2 at each stored draw: sampled values for `dIndependent_Normal_Gamma` components, constant columns for `dNormal` components, where the value was treated as known. `NULL` when `simulate = FALSE`.

`popgef.dispersion.mean` Named vector of posterior means of τ_k^2 (`colMeans(fixef.dispersion)`). `NULL` when `simulate = FALSE`.

`group.dispersion` Observation-level residual variance σ^2 : a scalar when `dispersion_ranef` is fixed, a length- n vector of final-sweep draws when `dispersion_ranef` is `dGamma()`; `NULL` when not applicable. When `simulate = FALSE`, the prior plug-in scalar (`prior$dispersion_ranef`) is returned for Gaussian models.

`group.dispersion.mean` Posterior mean of σ^2 ; equals `group.dispersion` when fixed. `NULL` when `group.dispersion` is `NULL`.

`draw_engine` Name of the Block~1 sampling engine used (e.g. `"rGLMM_sweeping_block1_ind"`). `NULL` when `simulate = FALSE`.

`fixef.iters` $n \times p_{re}$ matrix of the total number of Block~2 candidates generated per stored draw, summed over the `m_convergence` inner sweeps. `dIndependent_Normal_Gamma` components count envelope accept-reject candidates until acceptance (as `iters` in `glmbayes` samplers); `dNormal` components count exactly one conjugate draw per sweep. `NULL` when `simulate = FALSE`.

`fixef.iters.mean` Named vector of average candidates per accepted Block~2 draw (`colMeans(fixef.iters)/m_convergence`) equals 1 for `dNormal` components, and approximately the reciprocal acceptance rate of the ING envelope sampler otherwise). `NULL` when `simulate = FALSE`.

`fixef.mu` Numeric matrix $p_{re} \times J$ of Block 1 prior means at the final Gibbs state (from `build_mu_all`).

convergence List describing the sweep-count calibration: method ("exact", "exact_iid" when sim_method_used = "DEFAULT", or "local_gaussian_mode" for non-Gaussian [glmerb](#)), tv_tol, lambda_star, eigenvalues, m_min (derived minimum sweeps), and m_convergence (sweeps actually used). NULL when simulate = FALSE.

sim_method_used "DEFAULT" (exact iid draws) or "TWO_BLOCK_GIBBS" (two-block Gibbs sweeps), whichever engine actually ran. NULL when simulate = FALSE.

References

Nygren, K. (2020). *On the total variation distance between multivariate normal densities with applications to two-block Gibbs samplers*. Unpublished manuscript.

Jones, G. L. and Hobert, J. P. (2001). Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science* **16**, 312–334.

See Also

[Prior_Setup_GLMM](#), [model_setup](#), [build_mu_all](#), [two_block_rNormal_reg](#), [lmerb_posterior_mean](#), [rNormal_reg_group](#), [lmb](#), [glmb](#)

Examples

```
## lmerb: school random effects on bayesrules::big_word_club
##
## Random intercept and distracted_ppvt random slope by school; the school's
## share of free/reduced-lunch students (free_reduced_lunch, constant within
## school) is a level-2 predictor of the school intercepts.
##
## Fast man-page example: prior setup, lmer reference fit, and ICM posterior
## mean only (simulate = FALSE). No Gibbs draws – suitable for R CMD check.
##
## The full sampling workflows are preserved as demos:
## demo("Ex_14_lmerb_Sleepstudy", package = "lmebayes") # lme4-style small model
## demo("Ex_12_lmerb_BigWordClub", package = "lmebayes") # full formula

if (requireNamespace("bayesrules", quietly = TRUE)) {

  data(big_word_club, package = "bayesrules")
  dat <- big_word_club
  dat$school_id <- factor(dat$school_id)
  dat <- subset(dat, !is.na(score_ppvt) & !is.na(invalid_ppvt) &
    invalid_ppvt == 0L)
  dat <- dat[complete.cases(dat[, c("score_ppvt", "distracted_ppvt",
    "free_reduced_lunch", "school_id")]), ]

  form <- score_ppvt ~ free_reduced_lunch + distracted_ppvt +
    (1 + distracted_ppvt || school_id)

  ## Default hyperpriors calibrated from a reference lmer fit (weak prior).
  ps <- Prior_Setup_GLMM(form, data = dat, pop.pwt = 0.01)

  fit <- lmerb(
```

```

    form,
    data      = dat,
    pfamily_list = pfamily_list(ps),
    dispersion_ranef = ps$group.dispersion,
    simulate    = FALSE
  )
  ## lmer MLE vs ICM posterior mean (no MCMC means when simulate = FALSE).
  lmebayes::print_coef_means(fit)
  print(fit)
  summary(fit)
}

```

model_setup

Model setup for generalized linear mixed models

Description

Wrapper around `lmer` or `glmer` for models with exactly one grouping factor. Design matrices come from formula (including cross-level RE moderation terms). Random-effect and residual variance components (Ψ , dispersion) come from the same reference `lmer`/`glmer` fit on formula, so printed summaries match `summary(lmer(...))` on that formula.

Usage

```

model_setup(
  formula,
  data = NULL,
  family = gaussian(),
  REML = TRUE,
  control = NULL,
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  devFunOnly = FALSE,
  fit_mer = TRUE,
  dispformula = ~1,
  ...
)

```

Arguments

formula	A two-sided linear formula, as in <code>lmer</code> : response on the left of <code>~</code> , fixed and random-effects terms (<code>((terms group))</code>) on the right. Used for both design extraction and the reference <code>lmer</code> / <code>glmer</code> fit (fixed effects, hyper calibration,
---------	--

	and variance components); see Details for the single-grouping-factor / constraints.
data	An optional data frame containing the variables named in formula (as in <code>lmer</code>); if omitted, variables are taken from the environment of formula.
family	As in <code>glm</code> : a character string naming a family function, a family function, or the result of a call to one. Defaults to <code>gaussian()</code> , in which case <code>lmer</code> is used. Non-Gaussian families use <code>glmer</code> .
REML	Logical scalar, as in <code>lmer</code> 's own REML argument: should the reference fit's estimates optimize the REML criterion rather than the log-likelihood? Passed to <code>lmer</code> when <code>family = gaussian()</code> ; ignored otherwise (<code>glmer</code> has no REML criterion).
control	<code>lmerControl</code> when <code>family = gaussian()</code> , otherwise <code>glmerControl</code> – including the nonlinear optimizer and its settings (as in <code>lmer</code> / <code>glmer</code> 's own control argument); passed through to the reference fit when <code>fit_mer = TRUE</code> . NULL (default) resolves to <code>lmerControl()</code> 's defaults for Gaussian families, or <code>glmer</code> 's own default control otherwise.
start	Optional starting values, passed to the reference <code>lmer</code> / <code>glmer</code> fit's own start argument (numeric vector or named list with <code>par/theta</code> , and for <code>glmer</code> also <code>fixef/beta</code> ; see their documentation). NULL (default) uses <code>lme4</code> 's own defaults.
verbose	Integer scalar passed to the reference <code>lmer</code> (or <code>glmer</code> for non-Gaussian family); see their verbose argument (<code>> 0</code> prints optimizer iterations; <code>> 1</code> also prints inner PIRLS steps).
subset	An optional expression indicating the subset of rows of data to use, as in <code>lmer</code> 's subset argument. Passed to both <code>extract_re_hyper_matrices</code> (via <code>lFormula</code>) and the reference <code>lmer</code> / <code>glmer</code> (and <code>glmmTMB</code> when used) so the sampler design matrices and the classical reference fit share the same rows.
weights	An optional vector of prior weights, as in <code>lmer</code> 's weights argument. Passed to design extraction (<code>lFormula</code>), the reference <code>lmer</code> / <code>glmer</code> fit, and <code>glmmTMB</code> when used, so dropped rows match <code>subset/na.action</code> . Stored on the returned design as length-n weights (default all ones). Mixed-model sampler engines that still hard-code unit weights do not consume this vector yet; see <code>Prior_Setup_GLMM</code> .
na.action	A function indicating what should happen when data contain NAs (default <code>na.omit</code> , as in <code>lmer</code>). Passed to both design extraction (<code>lFormula</code>) and the reference fit so dropped rows match.
offset	An optional a priori known component to include in the linear predictor, as in <code>lmer</code> 's offset argument (plus any <code>offset()</code> term in formula). Passed to design extraction and the reference fit; stored on the returned design as length-n offset (default all zeros). Mixed-model sampler engines that still hard-code a zero offset do not consume this vector yet.
contrasts	An optional list of contrasts (see the <code>contrasts.arg</code> of <code>model.matrix.default</code> , as in <code>lmer</code>). Passed to both design extraction (<code>lFormula</code>) and the reference fit so factor coding in W/D matches <code>fixef/vcov</code> .
devFunOnly	Logical, as in <code>lmer</code> 's <code>devFunOnly</code> argument: if TRUE, return only the deviance evaluation function instead of a fitted model. Gaussian <code>lmer</code> fits only; ignored for <code>glmer</code> .

fit_mer	If TRUE (default), fit the reference lmer/glmer model and extract variance components – analogous to glmmTMB’s doFit argument. If FALSE, skip the reference fit and return design matrices and rank diagnostics only (used by glmerb() in lmebayes , which fits its own glmer separately).
dispformula	As in glmmTMB’s dispformula argument: a one-sided formula selecting the measurement-dispersion structure: ~1 (default, pooled) or ~<group_name> (matching the random-effects grouping factor exactly, requesting per-group dispersion). Only affects design\$glmmTMB_fit (see @return); design\$lmer/design\$glmer are always the plain pooled-dispersion lmer/glmer fit, regardless of dispformula. ~<group_name> additionally requires family = gaussian() (no observation-level dispersion parameter to model per group otherwise). Mirrors the dispformula argument on Prior_Setup_GLMM and on lmerb()/glmerb() in lmebayes ; all three are independent arguments that must be kept consistent by the caller.
...	Ignored.

Details

Uncorrelated random effects (||). The sampler treats Sigma_ranef as diagonal (no off-diagonal covariance). Multi-coefficient random terms must use ||, e.g. (1 + x || group) rather than (1 + x | group). A single random intercept may use (1 | group); (1 || group) is not supported by lme4.

Fixed-effect constraints. model_setup accepts the same formula language as [lmer](#), subject to one structural rule: every fixed effect that does *not* correspond to a random-slope term must be a *group-constant* (level-2) covariate—a predictor whose value is the same for every observation within a given group. School-level attributes such as private_school or title1 satisfy this constraint. Student-level covariates that vary *within* groups may appear as fixed main effects only when they *also* appear as random slopes (they then represent the population mean slope γ_{10} , e.g., distracted_ppvt). Cross-level interactions of the form level2_var:random_slope (e.g., free_reduced_lunch:distracted_a1) are additionally permitted; they moderate the prior mean of the corresponding random slope across groups (see [extract_re_hyper_matrices](#)). Fixed terms that are none of these three types—level-2 covariate, population mean slope, or cross-level moderation interaction—are rejected with an informative error.

Two-step identifiability assessment. After fitting lmer/glmer, model_setup calls [check_identifiability](#) to assess whether the model is empirically identified at both the within-group (groupef.rank, groupef.estimable, groupef.glm_check) and across-group (popef.rank, popef.deficient, popef.rank_ok) levels, and copies its fields onto the returned design object – see [check_identifiability](#) for the full algorithm description (including the family-specific estimability checks and how popef.rank_ok is derived). Non-estimable groups are flagged but retained in the lmer/glmer fit; [Prior_Setup_GLMM](#) excludes them when calibrating priors, and requires popef.rank_ok = TRUE to derive default hyperpriors automatically.

The example uses big_word_club from **bayesrules** (see ?bayesrules::big_word_club) and the same formula as the full lmerb() demo in lmebayes (demo("Ex_12_lmerb_BigWordClub", package = "lmebayes")).

Value

Object of class "model_setup" with:

y Response vector, length nrow(D).

weights Numeric prior-weight vector of length `length(y)` (ones when unspecified), aligned with `y/D/group` after `subset/na.action`.

offset Numeric offset vector of length `length(y)` (zeros when unspecified and no formula `offset()`), aligned with `y/D/group`.

D Group-effects model matrix (`n_obs x p_re`): per-observation loadings on the within-group group-effect coefficient vector (columns `groupef.names`).

group Factor of length `nrow(D)` for block subsetting; `levels(group)` fixes group order used elsewhere. Also carries `attr(group, "group_name")` (same value as `group_name`) so `group` alone can be passed to sampler engine functions without a separate `group_name` argument.

group_name Grouping factor name.

groupef.names Group-effect coefficient names from `lme4`; always identical to `names(W)` (and to `colnames(D)`).

W Named list of group-level hyper-design matrices (one per `groupef.names` entry, one row per group level).

popef.moderation Cross-level moderation metadata; see [extract_re_hyper_matrices](#).

formula The formula argument, as supplied.

family The normalized [family](#) object.

call The matched call.

terms The [terms](#) object for formula, as in `lm/glm` (and `glmbayes::glmb/lmb`).

data The data argument, as supplied (NULL when omitted), as in `lm/glm` (and `glmbayes::glmb/lmb`).

dispformula The `dispformula` argument, as supplied.

lmer / glmer The reference [lmer](#) (Gaussian) or [glmer](#) (otherwise) fit on formula; NULL when `fit_mer = FALSE`.

varcorr The raw [VarCorr](#) object from the reference fit.

Psi Named numeric vector (names `groupef.names`) of group-effect variances.

dispersion Residual (measurement) variance from the reference fit.

glmmTMB_fit NULL when `dispformula = ~1` or `fit_mer = FALSE`; otherwise a [glmmTMB](#) fit on formula with the same `dispformula` (Gaussian models only), fit via the same data/REML/ control-derived arguments as `lmer`. This is **additive**: `lmer/glmer` are never replaced by a `glmmTMB` fit. [Prior_Setup_GLMM](#) and `lmerb()` (in **lmebayes**) reuse `glmmTMB_fit` as their per-group-dispersion calibration reference instead of fitting `glmmTMB` a second time.

groupef.rank Named logical (names `levels(group)`): TRUE if that group's D_j is full column rank; see [check_identifiability](#).

groupef.estimable Named logical, same names as `groupef.rank`: TRUE if the group is additionally estimable (family-specific MLE/dispersion check); see [check_identifiability](#).

groupef.glm_check Per-group diagnostic data frame from [check_identifiability](#) (family-dependent; may be NULL).

popef.rank Named logical (names `groupef.names`): TRUE if `W[[k]]` is full column rank when restricted to `groupef.estimable` groups.

popef.deficient Negation of `popef.rank`.

popef.rank_ok Scalar TRUE only when every `popef.rank` entry is TRUE; required by [Prior_Setup_GLMM](#) to derive default hyperpriors automatically.

See Also

[check_identifiability](#), [extract_re_hyper_matrices](#)

Examples

```
## model_setup() - design + reference lmer (Ex_13b / big_word_club)
##
## Same formula and data cleaning as
## demo("Ex_13b_rLMM_estimated_dispersion_known_vcov_BigWordClub_PartVI").
## Requires suggested package bayesrules.

if (requireNamespace("bayesrules", quietly = TRUE)) {
  data(big_word_club, package = "bayesrules")

  dat <- big_word_club
  dat$school_id <- factor(dat$school_id)
  dat <- subset(
    dat,
    !is.na(score_ppvt) &
    !is.na(invalid_ppvt) & invalid_ppvt == 0L &
    complete.cases(dat[, c(
      "score_ppvt", "distracted_a1", "distracted_ppvt",
      "private_school", "title1", "free_reduced_lunch", "school_id"
    )])
  )

  form_lmer <- score_ppvt ~
    private_school + title1 + free_reduced_lunch +
    distracted_ppvt + distracted_a1 +
    free_reduced_lunch:distracted_a1 +
    (1 + distracted_ppvt + distracted_a1 || school_id)

  ## Default: design matrices + reference lmer + identifiability
  design <- model_setup(form_lmer, data = dat)
  print(design)

  ## Full-rank filter (same as Ex_13b before prior setup / sampling)
  full_rank <- names(design$groupef.rank)[design$groupef.rank]
  dat_fr <- subset(dat, school_id %in% full_rank)
  dat_fr$school_id <- droplevels(dat_fr$school_id)
  design_fr <- model_setup(form_lmer, data = dat_fr)
  stopifnot(all(design_fr$groupef.rank))
  print(design_fr)

  ## Variation: design only (no reference merMod fit)
  design_only <- model_setup(form_lmer, data = dat_fr, fit_mer = FALSE)
  stopifnot(is.null(design_only$lmer))
  names(design_only$W)
}
```

Description

Prior family objects provide a convenient way to specify the details of the priors used by matrix-input samplers such as `rglmb` and `rlmb`. See the documentation for `rglmb` and `rlmb` for the details of how such model fitting takes place.

Under a $\text{Beta}(\text{shape1}, \text{shape2})$ prior on the binomial probability θ and a $\text{Binomial}(n_i, \theta)$ likelihood with identity link ($\theta = \beta$ directly), the posterior is:

$$\theta \mid y \sim \text{Beta}(\text{shape1} + \sum n_i y_i, \text{shape2} + \sum n_i (1 - y_i)).$$

Usage

```
dNormal(mu, Sigma, dispersion = NULL)

dNormal_Gamma(mu, Sigma_0, shape, rate)

dIndependent_Normal_Gamma(
  mu,
  Sigma,
  shape,
  rate,
  max_disp_perc = 0.99,
  disp_lower = NULL,
  disp_upper = NULL
)

dGamma(
  shape,
  rate,
  beta,
  Inv_Dispersion = TRUE,
  lik_shape = 1,
  max_disp_perc = 0.99,
  disp_lower = NULL,
  disp_upper = NULL
)
```

Arguments

<code>mu</code>	a prior mean vector for the the modeling coefficients used in several pfamilies
<code>Sigma</code>	a prior variance-covariance matrix for <code>dNormal()</code> and <code>dIndependent_Normal_Gamma()</code> .
<code>dispersion</code>	the dispersion to be assumed when it is not given a prior. Should be provided when the Normal prior is for the <code>gaussian()</code> , <code>Gamma()</code> , <code>quasibinomial</code> , or

	quasipoisson families. The <code>binomial()</code> and <code>poisson()</code> families do not have dispersion coefficients. Omitted or NULL uses the internal default 1 and sets <code>ddef</code> in <code>prior_list</code> (see Details).
<code>Sigma_0</code>	prior variance-covariance on the precision-weighted coefficient scale for <code>dNormal_Gamma()</code> only (Gaussian). Stored in <code>prior_list\$Sigma</code> for compatibility with downstream samplers.
<code>shape</code>	The prior shape parameter for the gamma piece (inverse dispersion / precision). When taking defaults from Prior_Setup , use <code>ps\$shape</code> with <code>dNormal_Gamma()</code> and <code>dGamma()</code> , and <code>ps\$shape_ING</code> with <code>dIndependent_Normal_Gamma()</code> on the Gaussian calibrated path (see Details).
<code>rate</code>	The prior rate parameter paired with <code>shape</code> . With Gaussian Prior_Setup , <code>dNormal_Gamma()</code> and <code>dIndependent_Normal_Gamma()</code> use <code>ps\$rate</code> ; for <code>dGamma()</code> with fixed <code>beta</code> , prefer <code>ps\$rate_gamma</code> when that field is non-NULL (see Details).
<code>max_disp_perc</code>	Specifies the percentile used to truncate the posterior dispersion distribution when constructing the envelope for accept-reject sampling. This determines the lower and upper bounds for the dispersion (σ^2) used in the simulation. A value of 0.99 corresponds to using the central 98 percent of the posterior dispersion mass (i.e., excluding the outer 1 percent in each tail). Smaller values yield tighter bounds and may improve acceptance rates, while larger values allow broader dispersion support but may increase envelope complexity.
<code>disp_lower</code>	lower bound truncation for dispersion
<code>disp_upper</code>	upper bound truncation for dispersion
<code>beta</code>	Initial coefficient matrix (1×1); typically set to the prior mean <code>shape1/(shape1+shape2)</code> .
<code>Inv_Dispersion</code>	Logical (default TRUE). Controls which of the two Gamma prior roles <code>dGamma()</code> plays: • TRUE (default): prior on inverse dispersion (precision / shape parameter $k = 1/\phi$). This is the classical path used for dispersion estimation in Gaussian and Gamma(log) regression (<code>simfun = rGamma_reg</code>). • FALSE: conjugate prior on the Gamma or Poisson rate β directly (intercept-only, identity link). The posterior is a closed-form Gamma draw (<code>simfun = rGamma_Conjugate_reg</code>).
<code>lik_shape</code>	Known shape parameter $k > 0$ of the Gamma likelihood. Only used when <code>Inv_Dispersion = FALSE</code> and <code>family = Gamma(link = "identity")</code> . The intercept coefficient is then the Gamma rate β , and the conjugate posterior is $\beta \mid y \sim \text{Gamma}(\alpha_0 + nk, \beta_0 + \sum y_i)$. Defaults to 1 (exponential distribution). Ignored for Poisson families and whenever <code>Inv_Dispersion = TRUE</code> .

Details

`pfamily` is a generic with methods for fitted objects such as [rglmb](#) and [rlmb](#). The `dNormal()` prior is supported for all response families. The `gaussian()` family additionally supports `dNormal_Gamma()`, `dIndependent_Normal_Gamma()`, and `dGamma()` (precision prior). Intercept-only models with an identity link support two closed-form conjugate priors: `dBeta()` for `binomial(link = "identity")` and `dGamma(Inv_Dispersion = FALSE)` for `poisson(link = "identity")` and `Gamma(link = "identity")`.

A `pfamily` object represents a structured prior specification for use in Bayesian generalized linear modeling. Each constructor function (e.g., `dNormal()`, `dGamma()`, `dNormal_Gamma()`, `dBeta()`)

returns an object of class "pfamily" containing the prior parameters, supported likelihood families, compatible link functions, and a simulation function for posterior sampling.

These priors are designed to integrate seamlessly with modeling functions such as `rglmb()` and `rlmb()` in the **glmbayes** package, which consume the pfamily object to define the prior distribution over model parameters. The `pfamily()` generic retrieves the embedded prior from a fitted model object, while `print.pfamily()` displays its structure.

`prior_list`, `simfun`, and `pfun`. The named list `prior_list` holds the hyperparameters for the chosen prior family. When a model function draws from the posterior, it passes `prior_list` into the element `simfun` (e.g., `rNormal_reg`, `rGamma_reg`) so the low-level sampler receives one consistent list structure regardless of which constructor built the pfamily. Each constructor also stores a matching `pfun` (see `prior_simfuncs`) for i.i.d. draws from the prior via `pfun(n, prior_list, params)`, mirroring how `simfun` is used for the posterior.

Prior_Setup and default hyperparameters. `Prior_Setup()` fits an auxiliary GLM and returns default `mu`, `Sigma / Sigma_0`, `dispersion`, `Gamma` shape and rate, and related fields aligned with the data and prior-weight (`pwt`) choices. Those values can be supplied as arguments to the pfamily constructors when you want package-default priors on the same scale as the model matrix. Recommended use of shape and rate is not identical across constructors: for `dIndependent_Normal_Gamma()`, pass `shape = ps$shape_ING` from `Prior_Setup` (not the scalar `ps$shape` used by `dNormal_Gamma()`). For `dGamma()` with fixed coefficients (`beta`), pass `rate = ps$rate_gamma` when that field is present (otherwise `ps$rate`); see `Prior_Setup` and `compute_gaussian_prior`.

Prior Families:

- `dNormal()`: Specifies a multivariate normal prior over regression coefficients. It is conjugate for Gaussian likelihoods with an identity link function, and serves as the primary implemented prior for all other supported likelihood families in the current framework. This structure facilitates efficient posterior sampling and analytical tractability. The returned `prior_list` includes `ddef`: `TRUE` when dispersion was omitted or `NULL` (so the default 1 was used), `FALSE` when dispersion was supplied explicitly (including 1).
For models with log-concave likelihood functions—such as Poisson, Binomial, and Gamma families—posterior sampling under a `dNormal` prior is performed using a (Nygren and Nygren 2006) likelihood subgradient approach. This method constructs tight enveloping functions around the posterior using subgradients of the log-likelihood, enabling efficient accept-reject sampling even in high dimensions.
When the posterior distribution is approximately normal (typically the case for large sample sizes), the area under the enveloping function is bounded above by a constant factor—approximately $2/\sqrt{\pi} \approx 1.128$ in the univariate case, and $(2/\sqrt{\pi})^k$ in k -dimensional models. These bounds ensure that the rejection rate remains manageable and that the sampler remains computationally efficient.
The concept of conjugate priors was first formalized by (Raiffa and Schlaifer 1961), and further developed for regression models using g-prior structures by (Zellner 1986).
- `dGamma()`: A Gamma prior with two distinct roles controlled by `Inv_Dispersion`:
 - `Inv_Dispersion = TRUE` (default): prior on the inverse dispersion (precision $1/\phi$ or shape k). Used for dispersion estimation in Gaussian and Gamma(log) models, typically in a Gibbs step with `beta` held fixed (Gelman et al. 2013; Dobson 1990; McCullagh and Nelder 1989). With Gaussian `Prior_Setup` output, prefer `rate_gamma` for rate (see Details above).
 - `Inv_Dispersion = FALSE`: conjugate Gamma prior on the rate parameter β directly. Supports intercept-only models with an identity link: Poisson (Gamma–Poisson con-

jugacy) and Gamma (Gamma–Gamma conjugacy). Posterior draws are closed-form IID samples via `rGamma_Conjugate_reg`. The `lik_shape` argument specifies the known Gamma likelihood shape (default 1, i.e. exponential). `Prior_Setup` returns calibrated `conj_poisson` hyperparameters for this path.

- `dBeta()`: A Beta prior on the binomial probability θ for intercept-only binomial(`link = "identity"`) models. The posterior is a closed-form Beta draw (Beta–Binomial conjugacy) produced by `rBeta_reg`. Arguments `shape1` and `shape2` are the prior pseudo-success and pseudo-failure counts. `Prior_Setup` returns calibrated `conj_beta` hyperparameters for this path.
- `dNormal_Gamma()`: Combines a multivariate normal prior on coefficients with a gamma prior on precision, forming a conjugate structure for Gaussian models with unknown variance. The second argument is `Sigma_0` (precision-weighted scale); it is aliased internally to `Sigma` in `prior_list`. This formulation parallels classical Normal-Gamma models and is compatible with hierarchical extensions (Gelman et al. 2013; Raiffa and Schlaifer 1961).
- `dIndependent_Normal_Gamma()`: Similar to `dNormal_Gamma()`, but assumes independence between the coefficient and precision priors. This structure is useful for models where prior independence is desired or analytically convenient. With `Prior_Setup` on a Gaussian model, pass `shape_ING` as the `shape` argument (see Details above).

Each `pfamily` object includes:

- `pfamily`, `prior_list`, `okfamilies`, `plinks`, `simfun`, and `pfun` (see Value).

μ / Sigma : the surrogate Normal mean is $\text{shape1} / (\text{shape1} + \text{shape2})$ and the surrogate variance is the Beta variance $\text{shape1} * \text{shape2} / ((\text{shape1} + \text{shape2})^2 * (\text{shape1} + \text{shape2} + 1))$.

Value

An object of class "pfamily" (with a concise print method). A list with elements:

<code>pfamily</code>	Character string: the constructor name ("dNormal", "dGamma", "dNormal_Gamma", "dIndependent_Normal_Gamma", or "dBeta").
<code>prior_list</code>	Named list of prior hyperparameters. It is passed into <code>simfun</code> when sampling so the relevant low-level routine receives the prior in a fixed list form. Contents depend on the constructor: <code>dNormal</code> : <code>mu</code> , <code>Sigma</code> , <code>dispersion</code> , and logical <code>ddef</code> (TRUE if dispersion was omitted or NULL, so the default 1 was used; FALSE if set explicitly). <code>dGamma</code> : <code>shape</code> , <code>rate</code> , <code>beta</code> , <code>Inv_Dispersion</code> , <code>max_disp_perc</code> , <code>disp_lower</code> , <code>disp_upper</code> . When <code>Inv_Dispersion = FALSE</code> , also includes surrogate <code>mu</code> and <code>Sigma</code> (computed from the Gamma prior moments) and <code>lik_shape</code> . <code>dNormal_Gamma</code> : <code>mu</code> , <code>Sigma</code> (the <code>Sigma_0</code> precision-weighted input), <code>shape</code> , <code>rate</code> . <code>dIndependent_Normal_Gamma</code> : <code>mu</code> , <code>Sigma</code> (coefficient-scale covariance), <code>shape</code> , <code>rate</code> , <code>max_disp_perc</code> , <code>disp_lower</code> , <code>disp_upper</code> . <code>dBeta</code> : <code>shape1</code> , <code>shape2</code> , <code>beta</code> , and surrogate <code>mu</code> and <code>Sigma</code> computed from the Beta prior moments ($\mu = \text{shape1} / (\text{shape1} + \text{shape2})$, $\text{Sigma} = \text{shape1} * \text{shape2} / ((\text{shape1} + \text{shape2})^2 * (\text{shape1} + \text{shape2} + 1))$).
<code>okfamilies</code>	Character vector of implemented family names for which this <code>pfamily</code> may be used.
<code>plinks</code>	Function of one family argument returning allowed link names for that family.

simfun	Function used to generate posterior draws (e.g., rNormal_reg , rGamma_reg , rGamma_Conjugate_reg , rNormalGamma_reg , rindepNormalGamma_reg); for standard use these produce i.i.d.\ posterior samples for the implemented settings.
pfun	Prior-simulation function paired with this constructor (e.g., rNormal_prior , rGamma_prior , rNormal_Gamma_prior); called as <code>pfun(n, prior_list, params)</code> . See prior_simfuncs .

References

- Dobson A~J (1990). *An Introduction to Generalized Linear Models*. Chapman and Hall, London.
- Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2013). *Bayesian Data Analysis*, 3rd edition. CRC Press.
- McCullagh P, Nelder J~A (1989). *Generalized Linear Models*. Chapman and Hall, London.
- Nygren K~N, Nygren L~M (2006). "Likelihood Subgradient Densities." *Journal of the American Statistical Association*, **101**(475), 1144–1156. doi:[10.1198/016214506000000357](#).
- Raiffa H, Schlaifer R (1961). *Applied Statistical Decision Theory*. Clinton Press, Inc., Boston.
- Zellner A (1986). "On Assessing Prior Distributions and Bayesian Regression Analysis with g-Prior Distributions." In Goel P~K, Zellner A (eds.), *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti*, volume 6 of *Studies in Bayesian Econometrics and Statistics*, 233–243. Elsevier.

See Also

[rglmb](#), [rlmb](#) for modeling functions that consume pfamily objects.

[rNormal_reg](#), [rNormalGamma_reg](#), [rGamma_reg](#), [rGamma_Conjugate_reg](#), [rindepNormalGamma_reg](#) for lower-level posterior sampling functions used by pfamily constructors.

[prior_simfuncs](#) for the paired prior-simulation functions (pfun).

[Prior_Setup](#), [Prior_Check](#) for initializing and validating prior specifications.

[EnvelopeBuild](#) for envelope construction methods used in likelihood subgradient sampling (Nygren and Nygren 2006).

See also (Hastie and Pregibon 1992) for the original S modeling framework that inspired the design of pfamily.

pfamily_list

Build a named list of pfamily objects from a prior-setup object

Description

Generic for converting a prior-setup object into a named list of [pfamily](#) objects (class "pfamily_list") for block or mixed-model samplers.

Usage

```
pfamily_list(object, ptypes = NULL, ...)
```

Arguments

object	A prior-setup object. Dispatch is method-specific; see Methods and linked help pages for required classes and arguments.
ptypes	Prior-family name(s) per component; meaning depends on the active method (for row-block setup see Prior_SetupGroup).
...	Passed to methods (unused for Prior_SetupGroup).

Details

For a [Prior_Setup_GLMM](#) object, each group-effect coefficient k uses parameters from `object$pop.prior_list[[k]]`:

- "dNormal": `dNormal(mu, Sigma, dispersion)` with group-effect variance τ_k^2 from `pop.prior_list[[k]]$dispersion` (A12 §3.3.4 marginal hyper-regression center for Gaussian models; classical `VarCorr` in `pop.dispersion.ref`).
- "dIndependent_Normal_Gamma": the same `mu` and `Sigma`, plus a Gamma prior on precision $1/\tau_k^2$ from `object$pop.ing_prior[[k]]` (calibrated by [Prior_Setup_GLMM](#) via `pop.dispersion.pwt / pop.dispersion.nprior`, mean-matched at that same marginal τ_k^2). With effective prior sample size n_0 and p_k population predictors for component k ,

$$shape = (n_0 + 1 + p_k)/2, \quad rate = \tau_k^2 (n_0 + p_k - 1)/2.$$

Requires $n_0 \leq J$ (equivalently `pop.dispersion.pwt ≤ 0.5`). Truncation bounds `disp_lower / disp_upper` come from the limiting posterior window stored on `pop.ing_prior`; see `inst/ING_TRUNCATION_WINDOW`.

This is separate from observation-level σ^2 priors: those use [dGamma_list](#) (per-group `dispformula`) or a single [dGamma](#) built from `object$group.ing_prior` when dispersion is pooled.

Value

For `pfamily_list()`, an object of class "pfamily_list" (also inherits from "list"): a named list of pfamily objects. Attribute "ptypes" records the resolved prior-family name per component when the method sets it. For `print.pfamily_list()`, x invisibly.

See Also

[Prior_SetupGroup](#), [Prior_Setup_GLMM](#), [pfamily](#), [dNormal](#), [dNormal_Gamma](#), [dIndependent_Normal_Gamma](#)

plot_mean_convergence *Combined mean-convergence plot for Block~2 fixed effects (Claim 1)*

Description

Tracks convergence of the Block~2 fixed-effects (hyperparameter) posterior *mean* across inner Gibbs sweeps as a single combined chart (all traces overlaid on one axis), the mean-side counterpart of [plot_var_convergence](#)'s variance-side Claim~3 chart. Per the two-block Gibbs ergodicity result's Claim~1, the cross-chain mean bias $E[\gamma^{(l)}] - \gamma^*$ decays like A^l (linear in 1, vs. Claim~3's A^{2l} for variance). Each coefficient's per-sweep deviation from its reference mean is standardized by that same sweep's cross-chain SD, $Z(l) = (\text{mean}(l) - \text{mean}_{ref})/\text{sd}(l)$, so coefficients on very different scales can be overlaid on one chart and $Z(1)$ is expected to shrink toward 0 as 1 grows.

Usage

```
plot_mean_convergence(hist, ...)
```

Arguments

hist	Object of class "two_block_sweep_history", as carried by a fitted object's sweep_history (see print.two_block_sweep_history).
...	Passed to methods; see plot_mean_convergence.default .

Details

What is plotted:

The x-axis is the inner Gibbs sweep index l , and each point is a **cross-chain** statistic computed over the `n_chains` independent replicate chains, all of which sit at sweep l simultaneously. Nothing here is a running statistic along a single chain.

The plotted quantity is the standardized deviation

$$Z(l) = \frac{\text{mean}(l) - \text{mean}_{ref}}{\text{sd}(l)},$$

which divides out the scale of each coefficient so that an intercept measured in hundreds and a slope measured in tenths can share one axis. A converged run shows every trace scattered around 0 with no trend; a systematic drift toward or away from 0 means the sweep budget is too small. Supplying `n_chains` adds a naive band that separates Monte Carlo noise from genuine bias.

`mean_ref` is the exact posterior mean when design and `measurement_prior_list` are supplied (only possible when σ^2 and the RE variance components are all fixed), and otherwise the empirical last-sweep mean. As with [plot_var_convergence](#), the empirical fallback cannot detect a run that never converged, because the reference is drawn from the same unconverged distribution.

Reading it alongside the variance chart:

This chart is the location-side counterpart of [plot_var_convergence](#), which tracks spread. The variance converges at twice the rate of the mean, so this is the slower — and therefore the more informative — of the two when deciding whether `m_convergence` is large enough. A fit whose

variance ratios have settled at 1 while these traces still drift has converged in spread but not in location.

Because every replicate chain starts at the posterior mean, the bias begins at zero on Gaussian fits with fixed variance components: expect a flat chart from sweep 1 there. Read a non-flat one as evidence that something upstream — an estimated variance component, or a start at the mode rather than the mean — has moved the starting point off the target mean. `vignette("Chapter-B02", package = "lmebayesCore")` derives the two rates.

When there is nothing to plot:

As for `plot_var_convergence`: the exact route runs no sweeps to record, some Gaussian paths do not export per-sweep cross-chain statistics, and `whitened = TRUE` needs slots that only the sweep-recording engines populate. See that topic, which also explains how to obtain `n_chains` from a fitted object.

Value

hist invisibly.

See Also

`plot_var_convergence`, `lmerb_posterior_mean`

<code>plot_var_convergence</code>	<i>Combined variance-convergence plot for Block~2 fixed effects (Claim 3)</i>
-----------------------------------	---

Description

Tracks convergence of the Block~2 fixed-effects (hyperparameter) posterior variance across inner Gibbs sweeps as a single combined chart (all traces overlaid on one axis, following `lmebayes::Ex_16`'s style), per the two-block Gibbs ergodicity result (Claim~3 of the package's ergodicity reference, `inst/BLOCK_GIBBS_ERGODICITY.md`): after `l` sweeps, the cross-chain covariance $\Sigma_{11}^{(l)}$ satisfies

$$\Sigma_{11}^{-1/2} \Sigma_{11}^{(l)} \Sigma_{11}^{-1/2} = I - A^{2l}$$

for a convergent (symmetric, eigenvalues in $[0, 1)$) matrix A . `Var(1) / Var_ref` ratios are therefore bounded above by 1 and increase toward it as `l` grows.

Usage

```
plot_var_convergence(hist, ...)
```

Arguments

hist	Object of class "two_block_sweep_history" (see print.two_block_sweep_history). For whitened = TRUE, hist must also carry cov_by_sweep/coef_index, which are only populated by sweeps-outer/chains-inner engines – i.e. every engine driven by rGLMM_sweep : the <code>_run_with_pilot()</code> family (<code>rLMMNormal_reg_estimated_vcov()</code> , <code>rLMMIndepNormalGamma_reg_known_vcov()</code> , <code>rLMMIndepNormalGamma_reg_estimated_vcov()</code>) and, as of the <code>rGLMM_sweep()</code> -based rewrite of its engine, <code>rLMMNormal_reg_known_vcov(sim_method = "TWO_BLOCK_GIBBS")</code> as well.
...	Passed to methods (e.g. <code>coef_focus</code> , <code>design</code> , <code>whitened</code> , <code>n_chains</code> , ...; see below). Ignored by plot_var_convergence.default beyond the documented formals.

Details

What is plotted:

The x-axis is the inner Gibbs sweep index l , not the stored-draw index: these charts diagnose convergence *within* the burn-in that precedes each stored draw. Each point is a **cross-chain** statistic at sweep l — the variance across the `n_chains` independent replicate chains, all of which are at sweep l simultaneously — not a running statistic along a single chain. This is why the sweep-outer loop order matters: it is what makes "the sampler's distribution after l sweeps" an observable quantity.

The y-axis is the ratio $\text{Var}(1) / \text{Var}_{\text{ref}}$. Under Claim 3 that ratio is bounded above by 1 and rises toward it, so a converged run shows every trace flattening just below 1. Traces that are still climbing at the last sweep indicate the sweep budget is too small; traces that overshoot 1 substantially are Monte Carlo noise, and `n_chains` draws the band that says how much overshoot is expected.

`Var_ref` is the exact posterior variance when `design` and `measurement_prior_list` are both supplied (only possible when σ^2 and the RE variance components are all fixed), and otherwise the empirical last-sweep variance. The empirical fallback cannot detect a run that has not converged *at all*, since the denominator is then drawn from the same unconverged distribution; prefer the exact reference whenever the model allows it.

Reading it alongside the mean chart:

Location and spread converge at different speeds, and the two plotting functions separate them: [plot_mean_convergence](#) tracks the mean, [plot_var_convergence](#) the variance. The variance converges at twice the rate, so the mean chart settles later and is usually the one that decides whether the sweep budget is adequate. A run whose variance ratios sit at 1 while its standardized mean deviations still drift has converged in spread but not in location.

Because every chain starts at the posterior mean, the mean chart is flat from sweep 1 on well-behaved Gaussian fits, leaving the variance chart to carry the information. `vignette("Chapter-B02", package = "lmebayesCore")` derives the two rates.

When there is nothing to plot:

`sweep_history` is recorded only when all chains advance together, one sweep at a time, so that the cross-chain spread at sweep l is meaningful. Two situations produce no history:

- The exact route (fixed σ^2 , all components `dNormal`, `sim_method = "DEFAULT"`) runs no sweeps at all, because the draws are already exact. There is nothing to diagnose. Passing

`sim_method = "TWO_BLOCK_GIBBS"` forces the iterative path on the same target if you want a history anyway.

- Some Gaussian paths run sweeps but do not export per-sweep cross-chain statistics; `fit$sweep_history` is then `NULL`.

`whitened = TRUE` additionally requires `cov_by_sweep` and `coef_index`, which only the sweep-outer engines above populate, and `component = "precision"` additionally requires `disp_table`, which requires at least one ING Block 2 component.

Supplying `n_chains`:

`n_chains` is not stored on `hist` itself, so it must be supplied from the fitted object that produced it: for the main-stage history (`fit$sweep_history`) use `nrow(fit$popecf[[k]])` for any RE component `k` (all have the same number of rows, the `n` passed to the `_run_with_pilot()` call); for the pilot-stage history (`fit$pilot$draws$sweep_history`) use `fit$pilot$n`.

Value

`hist` invisibly.

See Also

[plot_mean_convergence](#), [lmerb_posterior_covariance](#)

<code>print.ranef.lmerb</code>	<i>Print random effects in lme4 layout</i>
--------------------------------	---

Description

Print random effects in **lme4** layout

Usage

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

Arguments

<code>x</code>	An object of class <code>"ranef.lmerb"</code> .
<code>...</code>	Ignored.

Value

`x`, invisibly.

Prior_Setup

*Setup Prior Objects***Description**

Helper function to facilitate the Setup of Prior Distributions for glm models.

Usage

```
Prior_Setup(
  formula,
  family = gaussian(),
  data = NULL,
  weights = NULL,
  subset = NULL,
  na.action = na.fail,
  offset = NULL,
  contrasts = NULL,
  pwt = NULL,
  pwt_default_low = 0.01,
  pwt_default_high = 0.05,
  n_prior = NULL,
  sd = NULL,
  dispersion = NULL,
  intercept_source = c("null_model", "full_model"),
  effects_source = c("null_effects", "full_model"),
  mu = NULL,
  k = 1,
  ...
)
```

Arguments

formula	an object of class " formula " (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	a description of the error distribution and link function to be used in the model.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>glm</code> is called.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process. (See additional details about how this argument interacts with data-dependent bases in the 'Details' below.)

na.action	how NAs are treated. The default is first, any <code>na.action</code> attribute of data, second a <code>na.action</code> setting of <code>options</code> , and third <code>na.fail</code> if that is unset. The factory-fresh default is <code>na.omit</code> . Another possible value is <code>NULL</code> .
offset	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector of length equal to the number of cases. One or more <code>offset</code> terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See <code>model.offset</code> .
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .
pwt	Weight on the prior relative to the likelihood function at the maximum likelihood estimate. If supplied, this value is used directly (scalar or one value per coefficient). If <code>n_prior</code> is provided and <code>pwt</code> is still a scalar and <code>sd</code> was not supplied, <code>pwt</code> is set to <code>n_prior / (n_prior + n_effective)</code> . If <code>length(pwt) > 1</code> (including from <code>sd</code>) or <code>sd</code> was supplied, <code>n_prior</code> does not overwrite <code>pwt</code> ; it is used only as a scalar for Gamma / S_{marg} steps. If <code>sd</code> is provided, <code>pwt</code> is computed from the prior standard deviations. If none of these are supplied, <code>pwt</code> defaults to <code>pwt_default_low</code> for models with fewer than 14 coefficients, and <code>pwt_default_high</code> otherwise.
pwt_default_low	Default prior weight used when <code>pwt</code> is not supplied and the model dimension is below 14. Defaults to 0.01.
pwt_default_high	Default prior weight used when <code>pwt</code> is not supplied and the model dimension is 14 or greater. Defaults to 0.05.
n_prior	Optional scalar effective prior sample size (on the <code>n_effective</code> scale). If provided with scalar <code>pwt</code> and without <code>sd</code> , <code>pwt</code> is recomputed from <code>n_prior</code> . With vector <code>pwt</code> or with <code>sd</code> , <code>pwt</code> is left unchanged and <code>n_prior</code> is used for the Gamma prior on precision and related Gaussian calibration only. If missing and <code>pwt</code> is scalar, <code>n_prior = (pwt/(1-pwt))*n_effective</code> .
sd	Optional vector argument with the prior standard deviations for the coefficients
dispersion	Optional scalar dispersion override (default <code>NULL</code>). For now, this is documented as an optional argument used to scale the Sigma (variance-covariance) matrix; see Details for additional context.
intercept_source	Specifies the method through which the prior mean for the intercept term is set. Options are based on the null intercept only model (<code>null_model</code>) or full_models. The default is the null model which is safer if variables are not centered.
effects_source	Specifies the method through which the prior means for the effects terms are set. Options are <code>null_effects</code> (prior means set to zero) or <code>full_model</code> (effect means set to match maximum likelihood estimates).
mu	Optional vector argument with the prior means for the coefficients
k	Scalar (default 1), non-negative ($k \geq 0$), with $k+p \geq 2$ where p is the number of coefficients (columns of the model matrix). k controls the tail behavior and effective degrees of freedom of the variance prior. It does not change the posterior mean of σ^2 or the covariance of β , but larger k makes the prior and posterior for σ^2 more concentrated and less heavy-tailed. Not yet used in calibration; passed through to <code>compute_gaussian_prior</code> for future use.

... For glm: arguments to be used to form the default control argument if it is not supplied directly.
 For weights: further arguments passed to or from other methods.

Details

Inputs to the function

The inputs to `Prior_Setup()` fall into three conceptual categories:

1. Model specification

- `formula`: structure of the GLM (response and predictors).
- `family`: error distribution and link.
- `data`, `weights`, `subset`, `na.action`, `offset`, `contrasts`, `control`, ...: as in `glm`.

2. Prior variance–covariance specification

- `pwt`: prior weight relative to the likelihood. If scalar, used to construct a Zellner-type g-prior. If vector, applied elementwise.
- `n_prior`: optional scalar effective prior sample size. Replaces scalar `pwt` only when `pwt` is scalar and `sd` is not used; otherwise supplies precision-prior / calibration only.
- `sd`: optional vector of prior standard deviations. If provided, used to compute `pwt` from the diagonal of `vcov(glm_full)`.
- `pwt_default_low`, `pwt_default_high`: defaults for `pwt` when not supplied.

3. Prior mean specification

- `intercept_source`: method for setting the prior mean of the intercept ("null_model" or "full_model").
- `effects_source`: method for setting the prior mean of the effects ("null_effects" or "full_model").
- `mu`: optional user-specified prior mean vector; overrides other centering logic if provided.

Prior covariance and Zellner scaling

Let $V_0 = \text{vcov}(\hat{\beta})$ be the covariance matrix of the full-model GLM coefficients. For non-Gaussian families, the prior covariance is:

$$\Sigma = \begin{cases} \frac{1 - \text{pwt}}{\text{pwt}} V_0, & \text{scalar pwt,} \\ V_0 \circ \left[\sqrt{\frac{1 - \text{pwt}_i}{\text{pwt}_i}} \sqrt{\frac{1 - \text{pwt}_j}{\text{pwt}_j}} \right], & \text{vector pwt,} \end{cases}$$

where $\circ$ denotes elementwise multiplication.

Intercept-only Poisson(link = "identity") conjugate prior on the rate

When the design is a single column (intercept only), `family = poisson()`, `link = "identity"`, scalar `pwt`, and offsets are zero, the effective conjugate prior observation count n_{prior} already satisfies $n_{\text{prior}} / (n_{\text{prior}} + n_{\text{effective}}) = \text{pwt}$ (otherwise `conj_poisson` remains NULL with a warning). Writing the weighted mean $\bar{y}_w = \sum_i w_i y_i / \sum_i w_i$, the output list component `conj_poisson` stores

shape = $n_{\text{prior}}\bar{y}_w$ and rate = n_{prior} , so the prior mean for the rate matches $\bar{y}_w$. Omitting optional mu resets the surrogate Normal summaries mu and the sole diagonal element of Sigma to these Gamma moments ($\text{Sigma}_{11} = \bar{y}_w/n_{\text{prior}}$).

For Gaussian families, Prior_Setup() also constructs the dispersion-free covariance

$$\Sigma_0 = \Sigma/\text{dispersion},$$

which under scalar pwt and the default calibration reduces to

$$\Sigma_0 = \frac{1 - \text{pwt}}{\text{pwt}} (X^\top W X)^{-1}.$$

Gaussian Normal–Gamma calibration and S_{marg}

For family = gaussian(), the function performs the Normal–Gamma calibration described in (Nygren 2025). Let:

- $p = \text{ncol}(x)$,
- $n_{\text{effective}} = \sum_i w_i$,
- $\hat{\beta}$ the weighted least-squares estimator,
- Σ_0 the dispersion-free prior covariance.

The marginal quadratic term is

$$S_{\text{marg}} = \text{RSS}_w + (\hat{\beta} - \mu)^\top (\Sigma_0 + (X^\top W X)^{-1})^{-1} (\hat{\beta} - \mu),$$

where RSS_w is the weighted residual sum of squares at $\hat{\beta}$. Under the default scalar-pwt Zellner mapping $\Sigma_0 = \frac{1 - \text{pwt}}{\text{pwt}} (X^\top W X)^{-1}$, this simplifies to

$$S_{\text{marg}} = \text{RSS}_w + \text{pwt} (\hat{\beta} - \mu)^\top (X^\top W X) (\hat{\beta} - \mu),$$

which makes the limiting behavior as $\text{pwt} \rightarrow 0$ transparent.

The calibrated dispersion is

$$\text{dispersion} = \frac{S_{\text{marg}}}{n_{\text{effective}} - p},$$

and the Normal–Gamma hyperparameters are

$$\text{shape} = \frac{n_{\text{prior}} + k}{2}, \quad \text{rate} = \frac{1}{2} S_{\text{marg}} \frac{n_{\text{prior}} + k + p - 2}{n_{\text{effective}} - p}.$$

The independent Normal–Gamma shape is

$$\text{shape}_{\text{ING}} = \text{shape} + \frac{p}{2}.$$

Posterior summaries for the conjugate Normal–Gamma prior

Under the conjugate Normal–Gamma prior (used by dNormal_Gamma()), the posterior has:

- Posterior mean

$$E[\beta \mid y] = (1 - \text{pwt}) \hat{\beta} + \text{pwt} \mu.$$

- Posterior expectation of σ^2

$$E[\sigma^2 | y] = \frac{S_{\text{marg}}}{n_{\text{effective}} - p}.$$

- Posterior covariance

$$\text{Cov}(\beta | y) = E[\sigma^2 | y] (\Sigma_0^{-1} + X^\top W X)^{-1}.$$

Weak-prior limits (Theorems 2 and 3)

As $n_{\text{prior}} \rightarrow 0^+$ (equivalently $\text{pwt} \rightarrow 0$), $S_{\text{marg}} \rightarrow \text{RSS}_w$, and the conjugate Normal–Gamma posterior converges to the classical weighted least-squares limit:

$$E[\beta | y] \rightarrow \hat{\beta}, \quad E[\sigma^2 | y] \rightarrow \frac{\text{RSS}_w}{n_{\text{effective}} - p}, \quad \text{Cov}(\beta | y) \rightarrow \frac{\text{RSS}_w}{n_{\text{effective}} - p} (X^\top W X)^{-1}.$$

For the independent Normal–Gamma prior used by `dIndependent_Normal_Gamma()`, neither the posterior mean nor the posterior covariance is available in closed form; the posterior must be obtained by numerical integration or sampling (e.g., `rindepNormalGamma_reg()`). Theorem 3 in (Nygren 2025) shows that the ING posterior has the same weak-prior limit as the conjugate Normal–Gamma posterior:

$$E[\beta | y] \rightarrow \hat{\beta}, \quad \text{Cov}(\beta | y) \rightarrow \frac{\text{RSS}_w}{n_{\text{effective}} - p} (X^\top W X)^{-1}.$$

Value

A list of class "PriorSetup" with components:

<code>mu</code>	Prior mean vector (length equal to the number of coefficients).
<code>Sigma</code>	Coefficient-scale prior variance–covariance matrix.
<code>Sigma_0</code>	For <code>family = gaussian()</code> only: dispersion-independent prior covariance on the precision-weighted coefficient scale (the <code>Sigma_0</code> passed to <code>compute_gaussian_prior</code>). Under scalar <code>pwt</code> , $\Sigma_0^{-1} = \frac{\text{pwt}}{1-\text{pwt}} X^\top W X$.
<code>dispersion</code>	Calibrated dispersion (Gaussian models only), equal to $S_{\text{marg}} / (n_{\text{effective}} - p)$ under the default calibration.
<code>shape</code>	Derived prior Gamma shape parameter for the Normal–Gamma prior on precision (Gaussian only), $(n_{\text{prior}} + k) / 2$.
<code>shape_ING</code>	For <code>gaussian()</code> only when <code>shape</code> is available: dedicated shape parameter for <code>dIndependent_Normal_Gamma()</code> , <code>shape + p / 2</code> .
<code>rate</code>	Derived prior Gamma rate parameter (Gaussian only), using the calibrated S_{marg} .
<code>rate_gamma</code>	For <code>gaussian()</code> only, when Gaussian calibration runs: prior Gamma rate for <code>dGamma()</code> / fixed- β use, based on $\text{RSS}_w(\beta_*)$ at the Zellner blend.
<code>coefficients</code>	Named numeric vector of returned coefficient values. For <code>gaussian()</code> with scalar or vector <code>pwt</code> , this is the closed-form posterior-mean blend $(1 - \text{pwt})\hat{\beta} + \text{pwt}\mu$ when inputs are valid; otherwise it falls back to the full-model GLM coefficients.
<code>model</code>	The model frame used to construct the design matrix (if <code>model = TRUE</code>).

x	The model matrix used (if x = TRUE).
y	The response vector used (if y = TRUE).
call	The matched call to Prior_Setup().
PriorSettings	A list containing prior configuration details, including pwt, n_prior, n_effective, n_likelihood, intercept_source, and effects_source.
conj_poisson	NULL unless family = poisson(link = "identity") with a single-column design (intercept-only), scalar pwt, finite positive scalar n_prior, and negligible offset. Then a named list with conjugate Gamma(shape, rate) hyperparameters on the Poisson rate, with shape = $n_{\text{prior}} \bar{y}_w$, rate = n_{prior} , and beta centered at the weighted sample mean $\bar{y}_w$; pass to dGamma with Inv_Dispersion = FALSE. See Details.

References

Nygren K (2025). "Core 10: Technical Derivations for Priors Returned by Prior_Setup." Vignette in the glmbayesCore R package. R vignette name: Core-10; formerly glmbayes appendix Chapter-A12.

See Also

[pfamily](#) for prior-family objects and the constructors [dNormal](#), [dNormal_Gamma](#), [dGamma](#), and [dIndependent_Normal_Gamma](#).

[rglmb](#), [rlmb](#) for matrix-input fits with a pfamily built from Prior_Setup() output; [rglmb](#), [rlmb](#) for matrix-based sampling that consumes the same prior structure; [simfuncs](#) for functions that take a prior_list assembled from those components (including [rindepNormalGamma_reg](#) for [dIndependent_Normal_Gamma\(\)](#)). [multi_prior_setup](#) for a matrix/cbind response with Gaussian; use with [rlmb](#) Prior_Setup per column.

(Zellner 1986); (Raiffa and Schlaifer 1961); (Gelman et al. 2013); (McCullagh and Nelder 1989); (Nygren 2025); (Nygren 2025).

Other prior: [Prior_Check\(\)](#), [multi_prior_setup\(\)](#)

Examples

```
## Not run:
## Full runnable examples are maintained in \pkg{glmbayesCore}:
example(Prior_Setup, package = "glmbayesCore", ask = FALSE, echo = TRUE)

## End(Not run)
```

Description

Calibrates priors for the level-2 fixed effects (fixef) of a hierarchical mixed model using the reference lmer/glmer fits on **all** groups (from [model_setup](#)). Per-group design rank (groupef.rank) and estimability (groupef.estimable – classical-glm MLE existence for family = binomial()/poisson()/Gamma(), residual degrees of freedom for family = gaussian()) are diagnostic checks computed by [model_setup](#) and do not subset the data or the reference glmer fit. Random-effect variances are treated as fixed at their mixed-model estimates. The returned object organizes calibrated priors as two independent blocks:

Usage

```
Prior_Setup_GLMM(
  formula,
  data,
  family = gaussian(),
  REML = TRUE,
  control = NULL,
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  dispformula = ~1,
  pop.pwt = NULL,
  pop.pwt_default_low = 0.01,
  pop.pwt_default_high = 0.05,
  pop.nprior = NULL,
  pop.sd = NULL,
  pop.dispersion.pwt = NULL,
  pop.dispersion.nprior = NULL,
  pop.max_disp_perc = 0.99,
  pop.intercept_source = c("null_model", "full_model"),
  pop.effects_source = c("null_effects", "full_model"),
  pop.mu = NULL,
  group.dispersion.pwt = NULL,
  group.dispersion.nprior = NULL,
  group.alpha_target = 0.01,
  group.max_disp_perc = 0.99
)
```

Arguments

formula	Mixed-model formula passed to model_setup , whose reference lmer/glmer fits (all groups) supply the calibration quantities.
data	Data frame containing all variables in formula.

family	Model <code>family</code> . Default <code>gaussian()</code> . Non-Gaussian families use <code>glmer</code> for calibration; Block~1 dispersion Gamma calibration is omitted (analogous to <code>Prior_Setup</code> for flat GLMs).
REML	Logical scalar passed to <code>model_setup</code> (and thus to the Gaussian reference <code>lmer</code> / <code>glmmTMB</code> fits). Default TRUE. Ignored for non-Gaussian families (<code>glmer</code> has no REML criterion). Affects the classical estimates used for prior calibration only.
control	Optional <code>lmerControl</code> / <code>glmerControl</code> object passed to <code>model_setup</code> . When NULL (default), Gaussian models use <code>lmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e5))</code> and non-Gaussian models leave <code>glmer</code> 's own control default.
start	Optional starting values passed to <code>model_setup</code> for the reference <code>lmer</code> / <code>glmer</code> fit. Default NULL.
verbose	Integer scalar passed to <code>model_setup</code> for the reference <code>lmer</code> / <code>glmer</code> optimizer. Default 0L.
subset, na.action	Optional arguments passed through to <code>model_setup</code> (same meanings as in <code>lmer</code> / <code>glmer</code>). <code>model_setup()</code> applies them to both design extraction (<code>lFormula</code>) and the reference <code>lmer</code> / <code>glmer</code> / <code>glmmTMB</code> fits, so sampler matrices (<code>design\$y/D/W</code>) and classical calibration use the same rows.
weights, offset	Optional arguments passed through to <code>model_setup</code> (same meanings as in <code>lmer</code> / <code>glmer</code>). Stored on <code>design\$weights</code> / <code>design\$offset</code> as length-n vectors aligned with <code>design\$y</code> (after <code>subset</code> / <code>na.action</code>), for Phase~1 availability to samplers. Not fully active for mixed-model sampling yet. They affect the classical reference fit used for prior calibration. Low-level <code>two_block_rNormal_reg</code> already accepts and echoes <code>prior.weights/offset2</code> when passed explicitly; high-level mixed routes that still hard-code unit weights / zero offset (and ICM / prior-weight algebra that assume unit weights) do not consume <code>design\$weights/design\$offset</code> yet. Prefer incorporating an offset in the formula or waiting until those paths are wired before relying on these for the Bayesian sampler.
contrasts	Optional contrast list passed to <code>model_setup</code> (as in <code>lmer</code>). Applied to both design extraction and the reference fit so factor coding in <code>W</code> matches <code>fixef/vcov</code> .
dispformula	One-sided formula selecting the Block~1 measurement- dispersion structure: <code>~1</code> (default, pooled) or <code>~<group_name></code> (matching the random-effects grouping factor exactly, requesting per-group dispersion). Passed through to <code>model_setup</code> , which fits and stores the <code>glmmTMB</code> reference (<code>design\$glmmTMB_fit</code>) reused below as <code>fit_ref</code> . Gaussian models only compute the per-group <code>group.ing_prior</code> calibration (a within-group regression fit for every group level, used only by <code>dGamma_list()</code>) when <code>dispformula</code> is the group formula; <code>dispformula = ~1</code> skips it entirely, so <code>group.dispersion.pwt</code> / <code>group.dispersion.nprior</code> must then be scalar (pooled). Mirrors the <code>dispformula</code> argument on <code>lmerb()</code> / <code>glmerb()</code> in <code>lmebayes</code> , which gates the analogous choice of sampler route; the two are independent arguments that must be kept consistent by the caller.
pop.pwt	Prior weight(s) in (0, 1) for the population-level (Block~2, <code>fixef</code>) coefficients. Either a scalar (applied to every random-effect component and every Block~2 predictor), or a list with one element per random-effect component (named with the RE coefficient names in any order, or unnamed positional). Each

list element is a scalar (recycled over that component's Block~2 predictors) or a vector of length p_k (optionally named with the predictor column names of `X_hyper[[k]]`, reordered to match). The prior covariance for each `fixef_k` block is scaled relative to `vcov(fit_ref)` following the [Prior_Setup](#) convention: $(1 - \text{pop.pwt}) / \text{pop.pwt}$ for a scalar, and elementwise $\sqrt{(1 - \text{pop.pwt}_i) / \text{pop.pwt}_i} \sqrt{(1 - \text{pop.pwt})}$ for vectors. At most one of `pop.pwt` (explicitly supplied), `pop.nprior`, and `pop.sd` may be supplied. When `pop.pwt` is NULL (the default), dimension-adaptive defaults are chosen separately for each random-effect component (see `pop.pwt_default_low` / `pop.pwt_default_high`).

`pop.pwt_default_low`

Scalar in $(0, 1)$, default 0.01 . Used when `pop.pwt` is NULL and a random-effect component's Block~2 predictor count $p_k = \text{ncol}(X_hyper[[k]])$ is strictly less than 14 (mirrors [Prior_Setup](#) `pwt_default_low`).

`pop.pwt_default_high`

Scalar in $(0, 1)$, default 0.05 . Used when `pop.pwt` is NULL and $p_k \geq 14$ for that component (mirrors [Prior_Setup](#) `pwt_default_high`).

`pop.nprior`

Optional *absolute* effective prior sample size(s), in group units ($J = \text{nlevels}(\text{group})$), for the population-level (Block~2, `fixef`) coefficients – an alternative to `pop.pwt`, on the same $n_i = J w_i / (1 - w_i)$ duality as `pop.dispersion.pwt` / `pop.dispersion.nprior`. Same shape as `pop.pwt`: a **scalar**, or a **list with one element per random-effect component** (named or unnamed positional), each element NULL (keep the `pop.pwt`-derived default for that component), a scalar, or a vector of length p_k (optionally named with the predictor column names of `X_hyper[[k]]`, reordered to match). Converted internally to an effective weight $w_i = \text{pop.nprior}_i / (\text{pop.nprior}_i + J)$, used in place of `pop.pwt` for that component's Sigma scaling. At most one of `pop.pwt` (explicitly supplied), `pop.nprior`, and `pop.sd` may be supplied. The returned `pop.nprior` field (see [Value](#) below) is always populated with the effective prior sample size actually implied (derived from whichever of `pop.pwt` / `pop.sd` was used when `pop.nprior` is not supplied), in the same shape this argument expects – call once without `pop.nprior` to inspect the defaults, then feed a modified copy of `ps$pop.nprior` back in to override.

`pop.sd`

Optional prior standard deviation(s), on the coefficient scale of `vcov(fit_ref)`, for the population-level (Block~2, `fixef`) coefficients – an alternative to `pop.pwt`. Same shape as `pop.pwt`: a **scalar**, or a **list with one element per random-effect component** (named or unnamed positional), each element NULL (keep the `pop.pwt`-derived default for that component – a *partial* override across components), a scalar, or a vector of length p_k (optionally named with the predictor column names of `X_hyper[[k]]`, reordered to match). Converted internally to an effective weight $w_i = V_{ii} / (V_{ii} + \text{pop.sd}_i^2)$, where V_{ii} is the corresponding diagonal entry of `vcov(fit_ref)`, and used in place of `pop.pwt` for that component's Sigma scaling. At most one of `pop.pwt` (explicitly supplied), `pop.nprior`, and `pop.sd` may be supplied. The returned `pop.sd` field (see [Value](#) below) is always populated with the effective standard deviations actually used ($\sqrt{\text{diag}(\Sigma_{\text{fixef}, k})}$, derived from whichever of `pop.pwt` / `pop.nprior` was used when `pop.sd` is not supplied), in the same shape this argument expects – call once without `pop.sd` to inspect the defaults, then feed a modified copy of `ps$pop.sd` back in to override.

pop.dispersion.pwt

Optional *relative* prior weight(s) in $(0, 1)$ for the Block~2 dispersion (precision) prior, decoupled from pop.pwt. A scalar, or a list / numeric vector with one value per random-effect component (named or positional). Converted internally to an effective prior sample size $n_k = J w_k / (1 - w_k)$ where J is the number of groups. At most one of pop.dispersion.pwt and pop.dispersion.nprior may be supplied; when neither is, the value is derived from pop.pwt (the mean across a component's predictors), keeping the Block~2 τ_k^2 dispersion prior consistent with the coefficient prior strength. Weak values carry no computational penalty for dIndependent_Normal_Gamma sampling: the τ^2 truncation window comes from limiting-posterior quantiles independent of the prior strength (see pop.ing_prior below).

pop.dispersion.nprior

Optional *absolute* effective prior sample size(s) (in group units) for the Block~2 τ_k^2 dispersion prior. A positive scalar, or a list / numeric vector with one value per random-effect component (named or positional). See pop.dispersion.pwt.

pop.max_disp_perc

Scalar in $(0.5, 1)$, default 0.99, or a named/positional length- p_{re} vector (one value per random-effect component). Same tail-probability role as group.max_disp_perc, but for the Block~2 τ_k^2 truncation window stored in pop.ing_prior. Only operationally consumed when the sampler's dispersion pfamily is dIndependent_Normal_Gamma (ignored for dNormal).

pop.intercept_source

Character string controlling the prior mean for the global intercept hyperparameter (Intercept)::(Intercept) only. One of "null_model" (default) or "full_model". When "null_model", the prior mean is taken from a random-intercept-only reference fit $y \sim 1 + (1 | \text{group})$ that omits all fixed-effect predictors (analogous to Prior_Setup with intercept_source = "null_model"). When "full_model", the full-model MLE intercept is used.

pop.effects_source

Character string controlling the prior mean for all other Block~2 hyperparameters (including population-mean slopes stored as (Intercept) columns in non-intercept RE components, and any non-intercept columns in X_hyper). One of "null_effects" (default) or "full_model". When "null_effects", prior means are set to zero. When "full_model", full-model MLE values are used.

pop.mu

Optional prior mean override(s) for the population-level (Block~2, fixed) coefficients – a finer-grained alternative to pop.intercept_source/pop.effects_source; prefer those first, and reach for pop.mu only when a specific mean needs to differ from either derived default. Same shape as pop.pwt: a **scalar**, or a **list with one element per random-effect component** (named or unnamed positional), each element NULL (keep the pop.intercept_source/pop.effects_source-derived default mean for that component – a *partial* override across components), a scalar, or a vector of length p_k (optionally named with the predictor column names of X_hyper[[k]], reordered to match). Replaces mu wholesale for the components supplied; components left NULL keep their usual pop.intercept_source/pop.effects_source default. The returned pop.mu field (see Value below) is always populated with the values actually used (derived defaults when pop.mu is not supplied), in the

same shape this argument expects – call once without `pop.mu` to inspect the defaults, then feed a modified copy of `ps$pop.mu` back in to override.

`group.dispersion.pwt`

Optional relative prior weight(s) in $(0, 1)$ for the Block~1 observation σ^2 Gamma prior (Gaussian models only), decoupled from `pop.pwt` (Block~2 `fixef`) and from `pop.dispersion.pwt` (Block~2 τ_k^2). Either a **scalar** converted to $n_{\text{prior}} = w/(1-w) \times n$ on the total observation count n (pooled `group.ing_prior`), or a **named or positional vector of length J** ($J = \text{nlevels}(\text{groups})$) with one weight per group level for per-group `dGamma_list()` calibration ($n_{\text{prior},j} = w_j/(1-w_j) \times n_j$). When `group.dispersion.pwt` is a vector, the pooled `group.ing_prior` continues to use the default $w = 0.01$ on total n (unless `group.dispersion.nprior` is supplied explicitly). At most one of `group.dispersion.pwt` and `group.dispersion.nprior` may be supplied; when neither is, scalar `group.dispersion.pwt = 0.01` is used for both pooled and per-group paths.

`group.dispersion.nprior`

Optional positive scalar: absolute effective prior sample size for the Block~1 σ^2 prior (observation units, not groups). See `group.dispersion.pwt`.

`group.alpha_target`

Scalar in $(0, 1)$, default 0.01 , or NULL to disable. Gaussian models with per-group `dispformula` only: target percentage of posterior `beta_j` draws allowed to fall outside the per-group Mahalanobis/log-concavity ellipsoid of `inst/BLOCK_GIBBS_ERGODICITY_IN` Section 16.6. This is a different quantity from `group.dispersion.pwt`: `group.dispersion.pwt` is the Gamma prior *weight* (an input to the prior), while `group.alpha_target` is a target *violation rate* (an output being aimed at). When non-NULL and `dispformula` requests per-group dispersion, `Prior_Setup_GLMM()` searches, for each group, for the smallest `group.dispersion.pwt` driving the predicted violation rate down to `group.alpha_target`, **floored at** the `group.dispersion.pwt` resolved from the `group.dispersion.pwt/group.dispersion.nprior` arguments above (so calibration only ever sharpens, never loosens, the per-group prior; supplying `group.dispersion.pwt` explicitly still guarantees a minimum). Active by default as a safeguard against poorly-behaved per-group priors; set to NULL to opt out and use `group.dispersion.pwt/group.dispersion.nprior` directly, unmodified. Silently ignored (not an error) when left at its default on a model that does not support per-group calibration (e.g. `dispformula = ~1`); it is only an error to *explicitly* supply a non-NULL value in that case. See `group.pwt_calibration` below.

`group.max_disp_perc`

Scalar in $(0.5, 1)$, default 0.99 , or a named/positional length- J vector (one value per group level, Gaussian models with per-group `dispformula` only). Tail probability used to compute the Block~1 σ^2 truncation window stored in `group.ing_prior` – the pooled scalar default when a vector is supplied here (`dispformula = ~1`), or the resolved per-group value (per group, per-group `dispformula`). The window is the central $2 \times \text{group.max_disp_perc} - 1$ mass interval: `disp_lower = (1 - group.max_disp_perc)` quantile and `disp_upper = group.max_disp_perc` quantile of the relevant Gamma (precision) distribution, inverted to the dispersion scale. Tighter values (e.g. 0.95) shrink the truncation window and typically improve Block~1 acceptance rates at the cost of slightly less envelope coverage; looser values (e.g. 0.999) widen the window. Passed to `dGamma()`

as `max_disp_perc`. A per-group vector lets outlier groups be given a tighter (or looser) window than the rest; see `dGamma_list()`.

Details

Block 1 (per-group, independent):

$$p(\mathbf{b}_j \mid \mathbf{y}, \text{fixef}, \sigma^2, \Sigma_b) = \mathcal{N}(\boldsymbol{\mu}_{b,j}^*, \Sigma_{b,j}^*)$$

$$\Sigma_{b,j}^{*-1} = \mathbf{Z}_j' \mathbf{Z}_j / \sigma^2 + \text{diag}(1/\tau_k^2)$$

when `family = gaussian()`. For non-Gaussian families there is no observation-level dispersion; Block~1 uses `dNormal` with `ddef = TRUE` (see [dNormal](#)).

Block 2 (per-RE coefficient k , independent):

$$p(\text{fixef}_k \mid \mathbf{b}_k, \tau_k^2) = \mathcal{N}(\boldsymbol{\mu}_{\text{fixef},k}^*, \Sigma_{\text{fixef},k}^*)$$

$$\Sigma_{\text{fixef},k}^{*-1} = \mathbf{X}_k' \mathbf{X}_k / \tau_k^2 + \Sigma_{\text{fixef},k}^{-1}$$

Naming convention. Top-level fields use `pop.*` for Block~2 (population / γ , τ_k^2) and `group.*` for Block~1 (within-group / observation σ^2 , RE covariance Ψ). Nested entries in `pop.prior_list[[k]]` use glmbayes-style `mu`, `Sigma`, and `dispersion` (where `dispersion` is the plug-in RE variance τ_k^2).

Why default calibration depends on classical estimates. `Prior_Setup_GLMM` scales Block~2 covariances from `vcov(fit_ref)` by $(1 - \text{pwt})/\text{pwt}$ and plugs in RE variances from the full reference fit, where `fit_ref` is the pooled lmer/glmer fit when `dispformula = ~1`, or an equivalent glmmTMB fit with the same `dispformula` when per-group dispersion is requested (see `calibration_source` and `mer_fit` above). By default the global intercept prior mean comes from a random-intercept-only null fit; all other prior means are zero (`pop.effects_source = "null_effects"`). This requires:

1. Converged reference lmer/glmer fit from [model_setup](#) on the full formula (and a random-intercept-only null fit when `pop.intercept_source = "null_model"`). Fits with lme4 `checkConv` failures (e.g. \large max|grad|) are rejected.
2. Every `X_hyper[[k]]` column maps to a `fixef(fit_ref)` term.
3. Each RE variance τ_k^2 from the reference fit is strictly positive.

Value

Object of class "Prior_Setup_GLMM" with fields:

`formula` Model formula.

`family` Family object.

`pop.pwt` Prior weight(s) used: the scalar as supplied, the canonical named list of per-predictor weight vectors when a list was supplied, or the dimension-adaptive default list (one weight per Block~2 predictor, per RE component) when `pop.pwt` was `NULL`.

`pop.nprior` Named list with one named numeric vector per random-effect component – the effective prior sample size (group units) actually implied, $n_i = J w_i / (1 - w_i)$, whether derived from `pop.pwt/pop.sd` (the usual case) or overridden via `pop.nprior`. Always present, and in the same shape the `pop.nprior` argument expects, so it can be inspected from a first call and fed back in (after editing) as `pop.nprior` to a second call.

- `pop.sd` Named list with one named numeric vector per random-effect component – the effective coefficient-scale prior standard deviations actually used ($\sqrt{\text{diag}(\Sigma_{\text{fixef},k})}$), whether derived from `pop.pwt/pop.nprior` (the usual case) or overridden via `pop.sd`. Always present, and in the same shape the `pop.sd` argument expects, so it can be inspected from a first call and fed back in (after editing) as `pop.sd` to a second call.
- `pop.mu` Named list with one named numeric vector per random-effect component – the prior means actually used (`mu`), whether derived from `pop.intercept_source/pop.effects_source` (the usual case) or overridden via `pop.mu`. Always present, and in the same shape the `pop.mu` argument expects, so it can be inspected from a first call and fed back in (after editing) as `pop.mu` to a second call.
- `pop.dispersion.pwt` Named per-component vector of relative dispersion prior weights (always present; consistent with `pop.dispersion.nprior` via $w_k = n_k / (n_k + J)$).
- `pop.dispersion.nprior` Named per-component vector of effective prior sample sizes for the Block~2 dispersion prior (always present; used by `pfamily_list()` to calibrate `dIndependent_Normal_Gamma` components).
- `group.dispersion.pwt` Gaussian models only: scalar or length- J vector of relative prior weights for Block~1 σ^2 – a scalar unless a per-group vector was supplied or `group.alpha_target` calibration was active, in which case it is the resolved length- J vector (see `dGamma_list()`).
- `group.dispersion.nprior` Gaussian models only: effective prior sample size for Block~1 σ^2 , on the same pooled-vs.-vector basis as `group.dispersion.pwt` (a scalar on the observation scale, or a named length- J vector of per-group $n_{\text{prior},j}$ reflecting the `group.alpha_target`-calibrated `group.dispersion.pwt` when calibration was active; see `group.pwt_calibration`).
- `group.alpha_target` The value used, or NULL if calibration was not active.
- `group.pwt_calibration` NULL unless calibration was active; otherwise a per-group diagnostics `data.frame` (`group`, `n_j`, `pwt_floor`, `pct_outside_before`, `w_star`, `clipped_at_ceiling`, `w_final`, `floor_binds`, `pct_outside_after`) from the search described under `group.alpha_target`.
- `group.max_disp_perc` Gaussian models only: scalar or length- J vector of the tail probability used for the Block~1 σ^2 truncation window (`group.ing_prior`) – the scalar as supplied, or the resolved length- J vector when a vector was supplied.
- `pop.max_disp_perc` Named length- p_{re} vector of resolved per-component values used for the Block~2 τ_k^2 truncation windows in `pop.ing_prior` (always present).
- `pop.intercept_source` How the global intercept prior mean was chosen ("null_model" or "full_model").
- `pop.effects_source` How other Block~2 prior means were chosen ("null_effects" or "full_model").
- `block_formula` Within-group Block~1 formula: response regressed on the random-coefficient structure only (columns of `design$group_efs.names`); level-2 hyper covariates are excluded. Used by `dGamma_list()`.
- `group.tau_sd` Named vector $\sqrt{\Psi}$ from the reference fit; shared population RE standard deviations for per-group calibration.
- `data` Data frame passed to `Prior_Setup_GLMM()` (reference for `dGamma_list()` diagnostics).
- `design` Full `model_setup` object (all groups), including length- n weights and offset vectors aligned with `design$y`.
- `mer_fit` Reference lmer/glmer fit on all groups (the full-formula fit from `model_setup`), always present regardless of `dispformula` (backs `group.dispersion` and `x$lmer/x$glmer` in `lmerb()/glmerb()`).

`fit_ref` The calibration reference for Block~2 (fixef/τ_k^2) and the per-group Block~1 inputs (`group.tau_sd`, BLUP coefficients): identical to `mer_fit` when `dispformula = ~1`; otherwise an equivalent glmmTMB fit with the same `dispformula` (Gaussian models only). See `calibration_source`.

`group.dispersion.fit` NULL when `dispformula = ~1`; otherwise the same glmmTMB object as `fit_ref`.

`group.dispersion.ref` NULL when `dispformula = ~1`; otherwise a named length- J vector of per-group observation-level dispersion read from `fit_ref`'s dispersion linear predictor (`glmmTMB::predict(fit_ref, type = "disp")`, aggregated by group). Diagnostic only – not the value fed to the sampler; compare against `group.ing_prior`'s `sigma2_hat`.

`calibration_source` "lme4" or "glmmTMB"; which reference fit produced `fit_ref` (and therefore `pop.prior_list`, `pop.ing_prior`, `group.tau_sd`, and `group.ing_prior`).

`group.dispersion` For Gaussian models: resolved Block~1 observation σ^2 used to mean-match the Gamma prior. When `dispformula = ~1`, the pooled A12 §3.3.4 aggregate $\hat{\sigma}^2 = (\sum_j S_{\text{marg},j}) / (n - Jp_{\text{re}})$ (Part VI Omega_j included); classical lmer residual in `attr(, "classical")`. When `dispformula` is per-group, a named length- J vector of per-group `group.ing_prior[[j]]$sigma2_hat`. `attr(, "source")` records provenance. NULL for non-Gaussian families. Pass to `rlmerb()/rglmerb()` as `group.dispersion` for fixed- σ^2 routes.

`group.Sigma` Diagonal RE covariance matrix (Block~1), classical reference-fit Ψ diagonals.

`pop.dispersion.ref` Named vector of classical reference-fit $\hat{\tau}_k^2$ (VarCorr / Psi) before the A12 §3.3.4 hyper-regression remargin; compare to `pop.prior_list[[k]]$dispersion`.

`pop.prior_list` Named Block~2 prior list per RE coefficient. For Gaussian models dispersion is the A12 §3.3.4 marginal hyper-regression center for τ_k^2 (not the raw VarCorr plug-in; see `pop.dispersion.ref`).

`pop.ing_prior` Named per-component list of the prospective dIndependent_Normal_Gamma calibration: Gamma precision-prior shape = $(n_0 + 1 + p_k)/2$ and rate = $\hat{\tau}_k^2(n_0 + p_k - 1)/2$ (the lmebayesCore default calibration with $n_0 = \text{pop.dispersion.nprior}$; since rate = $\hat{\tau}_k^2(\text{shape} - 1)$, the implied inverse-Gamma prior on τ_k^2 has mean exactly $\hat{\tau}_k^2$), where $\hat{\tau}_k^2$ is the A12 §3.3.4 marginal hyper-regression center in `pop.prior_list[[k]]$dispersion`, and the default τ_k^2 truncation window `disp_lower / disp_upper`: the $(1 - \text{pop.max_disp_perc}) / \text{pop.max_disp_perc}$ quantiles of the *limiting posterior* $\Gamma((J+1)/2, \hat{\tau}_k^2(J-1)/2)$ – the weak-prior ($n_0 \rightarrow 0$) limit of the Block~2 posterior Gamma for the precision (lmebayesCore Chapter A12, Theorem 2; inverted to a τ^2 interval). This window is identical for all n_0 , covers $\geq 2 \times \text{pop.max_disp_perc} - 1$ of the exact posterior for every prior strength, and keeps the envelope sampler's cost stable as priors weaken; see `inst/ING_TRUNCATION_WINDOW.md`. Used by `pfamily_list()` when `ptypes = "dIndependent_Normal_Gamma"`; ignored for dNormal priors.

`group.ing_prior` Gaussian models only; NULL for non-Gaussian models. Shape depends on `dispformula`:

`dispformula = ~1` (**pooled**) A single list: prospective dGamma() `group.dispersion` calibration for Block~1 ING (observation σ^2 shared across all group levels) – mean-matched shape / rate with $n_{\text{prior}} = \text{group.dispersion.pwt} / (1 - \text{group.dispersion.pwt}) \times n$, $p = p_{\text{re}}$, and $\hat{\sigma}^2 = \text{group.dispersion}$ (pooled A12 §3.3.4 aggregate of per-group $S_{\text{marg},j}$ by default; same ING algebra as `pop.ing_prior` for τ_k^2 ; requires `group.dispersion.pwt` ≤ 0.5), plus `disp_lower / disp_upper` as the central $2 \times \text{group.max_disp_perc} - 1$ prior-mass interval from the same calibrated shape/rate (always the pooled/scalar

group.max_disp_perc value, even when a per-group vector was supplied). Pass these fields to `dGamma()`.

dispformula requesting per-group dispersion A named list (one entry per group level) of per-group `dGamma()` density calibration (`sigma2_hat`, `shape_ING`, `rate`, `rate_gamma`, `n_prior`, `n_j`, `n_combined`, `disp_lower`, `disp_upper`, `max_disp_perc`, `omega_j`, ...). `shape_ING/rate` already fold in the Part VI model-derived `Omega_j` (fixed-effect/gamma uncertainty about `b_j`'s prior mean, propagated through group `j`'s hyper-design row and `pop.prior_list[[k]]$Sigma`), so this marginal integrates out both random effects (`b_j`) and fixed effects (`gamma`); see `inst/DGAMMA_LIST_MARGINAL_AND_BOUNDS.md` Part VI. `disp_lower/disp_upper` are $(1 - w_j)/w_j$ quantiles of `Gamma(shape_ING + n_j/2, rate_post)`, where `rate_post` is `sigma2_hat` mean-matched at that same inflated shape ($n_j/2$ added only for the window, so it reflects the posterior spread the sampler's own envelope machinery actually draws σ_j^2 from – see `EnvelopeDispersionBuild.cpp`'s `shape2 = Shape + n_w/2` fallback – while `shape_ING/rate` themselves, the values actually fed to the sampler as the `Gamma` prior, are unchanged); w_j is this group's own resolved group.max_disp_perc value (stored per-group as `max_disp_perc`). Used directly by `dGamma_list()`.

Which shape is returned is fully determined by `dispformula`; see `dGamma_list()` for how consumers tell the two apart.

`dispformula` The `dispformula` supplied.

`x` invisibly.

See Also

`model_setup`, `Prior_Setup`, `build_mu_all`, `pfamily_list`, `dGamma_list`

Examples

```
## Prior_Setup_GLMM() – Block~1 / Block~2 prior calibration (Ex_13b data)
##
## Same formula and cleaning as
## demo("Ex_13b_rLMM_estimated_dispersion_known_vcov_BigWordClub_PartVI").
## Requires suggested package bayesrules.

if (requireNamespace("bayesrules", quietly = TRUE)) {
  data(big_word_club, package = "bayesrules")

  dat <- big_word_club
  dat$school_id <- factor(dat$school_id)
  dat <- subset(
    dat,
    !is.na(score_ppvt) &
    !is.na(invalid_ppvt) & invalid_ppvt == 0L &
    complete.cases(dat[, c(
      "score_ppvt", "distracted_a1", "distracted_ppvt",
      "private_school", "title1", "free_reduced_lunch", "school_id"
    )])
  )
}
```

```

form_lmer <- score_ppvt ~
  private_school + title1 + free_reduced_lunch +
  distracted_ppvt + distracted_a1 +
  free_reduced_lunch:distracted_a1 +
  (1 + distracted_ppvt + distracted_a1 || school_id)

## Keep full-rank schools (sampler / per-group ING path)
design0 <- model_setup(form_lmer, data = dat)
dat <- subset(dat, school_id %in% names(design0$groupef.rank)[design0$groupef.rank])
dat$school_id <- droplevels(dat$school_id)

## Pooled measurement dispersion (dispformula = ~1, default)
ps_pooled <- Prior_Setup_GLMM(
  form_lmer,
  data = dat,
  pop.pwt = 0.01
)
print(ps_pooled)
names(ps_pooled$pop.prior_list)

## Per-group Block~1 calibration (Ex_13b-style dispformula).
## group.alpha_target = NULL skips the ellipsoid pwt search (faster for examples);
## Ex_13b uses group.alpha_target = 0.01 with group.dispersion.pwt = 0.1 as floor.
ps_group <- Prior_Setup_GLMM(
  form_lmer,
  data = dat,
  pop.pwt = 0.01,
  dispformula = ~school_id,
  group.max_disp_perc = 0.8,
  group.dispersion.pwt = 0.1,
  group.alpha_target = NULL
)
print(ps_group)
stopifnot(is.list(ps_group$group.ing_prior[[1L]]))
length(ps_group$group.ing_prior)
}

```

Prior_SetupGroup

Prior setup for row-block regressions

Description

Runs [Prior_Setup](#) on each block subset of the data. Typical callers are [rNormal_reg_group](#) / [rNormalGLM_reg_group](#) or the `lmbBlock` / `glmbBlock` wrappers in **lmebayes**.

Usage

```

Prior_SetupGroup(
  formula,
  group,

```

```

family = gaussian(),
data = NULL,
weights = NULL,
subset = NULL,
na.action = na.fail,
offset = NULL,
contrasts = NULL,
pwt = NULL,
pwt_default_low = 0.01,
pwt_default_high = 0.05,
n_prior = NULL,
sd = NULL,
dispersion = NULL,
intercept_source = c("null_model", "full_model"),
effects_source = c("null_effects", "full_model"),
mu = NULL,
k = 1,
...
)

```

Arguments

formula	A formula with a single response.
group	Group partition: factor or vector of length <code>nrow(data)</code> (after <code>model.frame</code>), a column name in <code>data</code> , <code>l2_blocks</code> counts, or a list of row index vectors (see normalize_group).
family	a description of the error distribution and link function to be used in the model.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in <code>data</code> , the variables are taken from <code>environment(formula)</code> , typically the environment from which <code>glm</code> is called.
weights	an optional vector of ‘prior weights’ to be used in the fitting process. Should be <code>NULL</code> or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process. (See additional details about how this argument interacts with data-dependent bases in the ‘Details’ below.)
na.action	how NAs are treated. The default is first, any na.action attribute of <code>data</code> , second a <code>na.action</code> setting of options , and third <code>na.fail</code> if that is unset. The factory-fresh default is <code>na.omit</code> . Another possible value is <code>NULL</code> .
offset	this can be used to specify an <i>a priori</i> known component to be included in the linear predictor during fitting. This should be <code>NULL</code> or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset .
contrasts	an optional list. See the <code>contrasts.arg</code> of <code>model.matrix.default</code> .

pwt, n_prior, sd, mu, dispersion, k

Each of these six calibration arguments (passed to `Prior_Setup` for every block) may be supplied in **either** of two forms:

- the ordinary shared form documented in `Prior_Setup` (a scalar, or a per-coefficient vector for pwt/sd/mu) – applied identically to every block (default); or
- a **named list with one element per block**, keyed by the block/group level IDs (`names(Prior_SetupGroup(...))`, i.e. `normalize_group()$ids`) – each element supplies that block’s own value, in the same shape `Prior_Setup()` would otherwise accept. `names(x)` must exactly match the block IDs. An element may be `NULL`, meaning use `Prior_Setup()`’s default for that argument in that block.

The two forms may be mixed freely across the six arguments in one call.

pwt_default_low

Default prior weight used when pwt is not supplied and the model dimension is below 14. Defaults to 0.01.

pwt_default_high

Default prior weight used when pwt is not supplied and the model dimension is 14 or greater. Defaults to 0.05.

intercept_source

Specifies the method through which the prior mean for the intercept term is set. Options are based on the null intercept only model (`null_model`) or `full_models`. The default is the null model which is safer if variables are not centered.

effects_source

Specifies the method through which the prior means for the effects terms are set. Options are `null_effects` (prior means set to zero) or `full_model` (effect means set to match maximum likelihood estimates).

...

Not used.

Details

What it produces:

`Prior_SetupGroup()` calls `Prior_Setup` once per block, on that block’s rows only, and returns the results as a named list keyed by block ID.

Contrast with hierarchical mixed-model setup:

`Prior_Setup_GLM` calibrates a **hierarchical** prior (one reference lmer/glmer fit, Block~2 hyperpriors, cross-group shrinkage). `Prior_SetupGroup()` calibrates **independent** priors per row block with no shared γ or τ^2 ; row-block engines draw blocks iid.

Names for per-block calibration lists must match `normalize_group` block IDs exactly.

Prior families:

After calibration, convert to sampler-ready `pfamily` objects with `pfamily_list(x, ptypes = ...)` (see **Prior families via** `pfamily_list()` below for `ptypes` and allowed families).

Value

A named list of class "Prior_SetupGroup". Each element is a [Prior_Setup](#) result for one block.
 x invisibly.

An object of class "pfamily_list": a named list of pfamily objects with attribute "ptypes".

See Also

[Prior_Setup](#), [multi_prior_setup](#), [normalize_group](#), [pfamily_list](#), [rNormal_reg_group](#), [rNormalGLM_reg_group](#),
[Prior_Setup_GLMM](#)

ranef.lmerb

Extract random effects from Bayesian mixed model fits

Description

Methods mirroring [ranef](#) for [lmerb](#) and [glmerb](#) fits. Values summarize Block~1 random-effects draws b stored in `fit$groupef` (posterior chain mean by default, or the exact ICM `fit$groupef.mode`).

Usage

```
## S3 method for class 'lmerb'
ranef(object, condVar = FALSE, drop = FALSE, type = c("mean", "mode"), ...)

## S3 method for class 'glmerb'
ranef(object, condVar = FALSE, drop = FALSE, type = c("mean", "mode"), ...)
```

Arguments

object	An lmerb or glmerb object.
condVar	Logical; if TRUE, attach a postVar attribute (posterior variance from stored draws, diagonal by RE term).
drop	Logical; if TRUE, convert single-column grouping data.frames to named vectors (as in lme4).
type	"mean" (default when draws are stored) or "mode" (ICM / groupef.mode). When simulate = FALSE, "mean" falls back to "mode" with a warning.
...	Ignored.

Value

An object of class "ranef.lmerb": a named list with one data.frame per grouping factor (rows = levels, columns = RE terms). Attributes include type, n_draws, and optionally postVar.

See Also

[ranef](#), [coef.lmerb](#), [fixef.lmerb](#), [lmerb](#).

Description

rglmerb draws from the posterior of a Bayesian generalized linear mixed model — Poisson, binomial, Gamma, or Gaussian — with a single grouping factor and uncorrelated random effects, given a design from [model_setup](#) and priors for the population level (pfamily_list) and, for Gaussian responses only, the residual variance (dispprior_list).

Usage

```
rglmerb(
  n,
  design,
  pfamily_list,
  family = poisson(),
  dispprior_list = NULL,
  offset = NULL,
  weights = 1,
  gap_tol = 0.0196,
  tv_tol = 0.01,
  mode_gap_max = 1,
  collect_block1 = TRUE,
  verbose = TRUE,
  progbar = FALSE,
  sim_method = "DEFAULT",
  simulate = TRUE
)
```

Arguments

n	Integer. Number of independent chains in the main stage. Required when simulate = TRUE; ignored when simulate = FALSE.
design	A model_setup object (from model_setup).
pfamily_list	Named list of Block~2 pfamily objects, one per random-effect coefficient in design\$groupef.names (same as rGLMM_reg / rLMMNormal_reg).
family	A family object. Default poisson().
dispprior_list	Block~1 observation-dispersion prior, in the same shapes as the rLMM* / rGLMM* engines. Required for family = gaussian() (list(dispersion =) or dGamma() / dGamma_list()); must be NULL (default) for poisson() and binomial().
offset, weights	Observation offset and prior weights, as in rGLMM_reg / rLMMNormal_reg (offset = NULL, weights = 1). When omitted, inherit design\$offset / design\$weights if present. Explicit values always override the design. Echoed on the fit; not yet used in Gibbs sweeps.

gap_tol	Legacy mode—mean gap for deriving the pilot chain count when tv_tol is NULL. Ignored for Gaussian without ING population components. Ignored when simulate = FALSE.
tv_tol	Total variation tolerance for convergence calibration. Inner Gibbs sweeps and pilot chain counts are derived internally. Ignored when simulate = FALSE.
mode_gap_max	Pilot inner-sweep calibration (non-Gaussian and Gaussian+ING only). Ignored when simulate = FALSE.
collect_block1	Collect group-level draws from main chains (non-Gaussian only). Ignored when simulate = FALSE.
verbose	Print stage headers and diagnostics.
progbar	Progress bars when verbose is FALSE. Ignored when simulate = FALSE.
sim_method	Simulation method for family = gaussian(): "DEFAULT" or "TWO_BLOCK_GIBBS"; see rlmerb . Ignored (two-block Gibbs only) for non-Gaussian families. Ignored when simulate = FALSE.
simulate	Logical (default TRUE). When TRUE, draw posterior samples. When FALSE, return point estimates only via rglmerb_point (no MCMC).

Details

The returned sample is n mutually independent draws, one per replicate chain, taken at that chain's final sweep. It is analyzed like an iid sample rather than like an MCMC trace.

This is the matrix-level interface; for a formula interface use `glmerb()` in **lmebayes**. [rlmerb](#) is the Gaussian-only counterpart.

The model being estimated:

`rglmerb` estimates the same two-stage structure as [rlmerb](#), with a link function between the linear predictor and the response mean. For group j with link g :

$$y_j \mid \beta_j \sim \text{family}(g^{-1}(D_j \beta_j)) \quad (\text{within group})$$

$$\beta_j \mid \gamma, \Psi \sim N(\mathcal{W}_j \gamma, \Psi) \quad (\text{across groups})$$

The across-group stage is unchanged and remains Gaussian: group coefficients scatter around a population expectation $\mathcal{W}_j \gamma$ with diagonal covariance $\Psi = \text{diag}(\tau_k^2)$. What changes is the within-group stage and, with it, the interpretation of β_j , which now lives on the link scale — log rates for `poisson()`, log-odds for `binomial()`. Summaries that are meaningful on the response scale need the inverse link applied to the draws.

Supported families are `gaussian()`, `poisson()`, `binomial()`, and `Gamma()` with their standard links. The restriction is log-concavity of the likelihood, which the sampler requires (Nygren and Nygren 2006).

What you supply:

design A [model_setup](#) object built with the matching family. For non-Gaussian responses its estimability report matters more than usual: a group with no variation in the response (all-zero counts, a single binomial outcome level) has no finite maximum likelihood estimate, and is reported separately from the rank check. See `vignette("Chapter-B08", package = "lmebayesCore")`.

pfamily_list The population-level prior, one component per random-effect coefficient, declaring whether each between-group variance τ_k^2 is known (`dNormal`) or estimated (`dIndependent_Normal_Gamma`).
disprior_list Only meaningful for `gaussian()`, where it gives the residual variance. `poisson()` and `binomial()` have no free dispersion parameter, so it must be `NULL`.

What the returned draws are:

As in `rlmerb`, `rglmerb` runs n independent replicate chains and stores one draw from each, taken at the final sweep, so the n draws are mutually independent and ordinary Monte Carlo standard errors apply. There is nothing to thin and no autocorrelation to diagnose.

The strength of the accompanying guarantee is weaker here than in the Gaussian case, and the difference is worth understanding rather than glossing over. The calibration behind `m_convergence` is derived for Gaussian targets. A Poisson or binomial posterior is not Gaussian, so the engine applies the same calibration at curvature evaluated near the posterior mode. The sweep count is therefore a well-motivated **budget** rather than a certificate, and `tv_tol` should be read in that spirit. Only `family = gaussian()` with fixed σ^2 and all-`dNormal` components yields exact draws.

A second difference follows from the same fact. Because the posterior mode is not the posterior mean when the target is skewed, chains started at the mode begin at an unknown distance from the target. `rglmerb` therefore runs a short **pilot stage** first and uses its cross-chain mean as the starting point for the main chains, which shortens the main stage. Pilot draws are diagnostic only and are discarded — they are never part of the returned sample. `vignette("Chapter-B03", package = "lmebayesCore")` covers the trade-off between pilot size and main-stage length.

Point estimates without sampling:

`simulate = FALSE` returns the joint posterior mode by iterated conditional modes (or the exact mean for `gaussian()`), at fixed variance-component plug-ins. No sweeps run and no RNG is consumed, so the result is deterministic — useful while iterating on model specification. All draw, pilot, and convergence slots are `NULL`.

Value

An object of class `c("rglmerb", "list")`, with the same layout as `rlmerb`. Each draw slot holds one row per replicate chain, taken at that chain's final sweep.

popef Named list of population draws γ , one $n \times q_k$ matrix per random-effect coefficient, on the link scale. `popef.means` and `popef.mode` are the draw mean and the posterior mode.

groupef Long-format data frame of group coefficients β_j ($n \times J$ rows: draw, the grouping factor, then one column per coefficient), also on the link scale. `groupef.mode` is the $J \times p_{re}$ mode matrix. Collected only when `collect_block1 = TRUE`.

popef.dispersion Draws of the between-group variances τ_k^2 ; constant for `dNormal` components, varying for estimated ones. `popef.dispersion.mean` is the column mean.

group.dispersion Residual variance, Gaussian responses only; absent for `poisson()` and `binomial()`.

m_convergence, convergence_info Sweeps run per chain and the calibration behind that count (`method`, `lambda_star`, `draw_engine`, `sim_method_used`, `icm_info`).

pilot Present when a pilot stage ran, holding its size n , the `chisq` calibration constant, and its draws. Diagnostic only — these are not posterior draws.

Inputs are echoed back in `design`, `family`, `n`, `call`, `pfamily_list`, `dispprior_list`, `prior.weights` and `offset`, alongside the packed prior and thin Prior summaries.

When `simulate = FALSE`, every draw, pilot and convergence slot is NULL and only the mode fields are populated.

See Also

[r1merb](#), [rglmerb_point](#), [rLMMNormal_reg_known_vcov](#), [rLMMNormal_reg_estimated_vcov](#), [rLMMindepNormalGamma_reg_known_vcov](#), [rLMMindepNormalGamma_reg_estimated_vcov](#), [rGLMM_reg](#), [Prior_Setup_GLMM](#)

r1merb

The Bayesian Linear Mixed-Effects Model Distribution

Description

`r1merb` draws from the posterior of a Bayesian linear mixed model with a single grouping factor and uncorrelated random effects, given a design from [model_setup](#) and priors for the population level (`pfamily_list`) and the residual variance (`dispprior_list`).

Usage

```
r1merb(
  n,
  design,
  pfamily_list,
  dispprior_list,
  offset = NULL,
  weights = 1,
  tv_tol = 0.01,
  progbar = TRUE,
  verbose = TRUE,
  print_icm_table = TRUE,
  gap_tol = 0.0196,
  mode_gap_max = 1,
  diag_sweeps = FALSE,
  sim_method = "DEFAULT",
  simulate = TRUE
)
```

Arguments

<code>n</code>	Integer. Number of stored draws (each draw is one full pass through <code>m_convergence</code> inner Gibbs sweeps). Required when <code>simulate = TRUE</code> ; ignored when <code>simulate = FALSE</code> .
<code>design</code>	A <code>model_setup</code> object (from model_setup) supplying <code>y</code> , <code>D</code> , <code>group</code> , <code>W</code> , <code>group_name</code> , and <code>groupec.names</code> .

<code>pfamily_list</code>	Named list of Block~2 pfamily objects, one per random-effect coefficient in <code>design\$groupef.names</code> (same as <code>rLMMNormal_reg</code>).
<code>disprior_list</code>	Block~1 observation-dispersion prior, in the same shapes accepted by the <code>rLMM*</code> engines: <code>list(dispersion=)</code> for fixed σ^2 (scalar or per-group vector); a <code>dGamma()</code> pfamily (pooled ING); or a named list of <code>dGamma()</code> pfamilies (per-group ING). A bare positive numeric scalar/vector is also accepted (treated as fixed dispersion).
<code>offset, weights</code>	Observation offset and prior weights, as in <code>rLMMNormal_reg</code> (<code>offset = NULL</code> , <code>weights = 1</code>). When omitted, inherit <code>design\$offset / design\$weights</code> from <code>model_setup</code> if present. Explicit values always override the design. Echoed on the fit as <code>prior.weights / offset / offset2</code> ; not yet used in Gibbs sweeps.
<code>tv_tol</code>	Single numeric in $(0, 1)$. Total variation tolerance used for convergence calibration. Default 0.01. Inner Gibbs sweeps per stored draw are derived from Theorem~3. Ignored when <code>simulate = FALSE</code> .
<code>progbar</code>	Logical. Show a text progress bar during sampling. Default TRUE. Ignored when <code>simulate = FALSE</code> .
<code>verbose</code>	Logical. Print the reference-vs-ICM table and the convergence calibration line. Default TRUE.
<code>print_icm_table</code>	Logical. When FALSE, skip the reference-vs-ICM table. The convergence calibration line from the Core sampler still follows verbose. Default TRUE.
<code>gap_tol</code>	Legacy mode—mean gap tolerance for the pilot stage when any population component uses <code>dIndependent_Normal_Gamma</code> and <code>tv_tol</code> is NULL. Ignored for all-dNormal models. Ignored when <code>simulate = FALSE</code> .
<code>mode_gap_max</code>	Pilot inner-sweep calibration for ING population models (default 1.0). Ignored for all-dNormal models. Ignored when <code>simulate = FALSE</code> .
<code>diag_sweeps</code>	Diagnostic flag for ING population models with a pilot stage. When TRUE, auto-print one combined population chain-mean table per stage when each stage finishes; <code>sweep_history</code> is always stored on the fit. Default FALSE. Ignored when <code>simulate = FALSE</code> .
<code>sim_method</code>	Simulation method: "DEFAULT" or "TWO_BLOCK_GIBBS". Only affects the fixed-dispersion, known-variance-components route (scalar or per-group fixed <code>disprior_list</code> , all population <code>dNormal()</code>): "DEFAULT" draws directly from the exact multivariate-normal posterior via <code>rLMMNormal_reg_known_vcov_iid</code> (no Gibbs sweeps, no burn-in); "TWO_BLOCK_GIBBS" forces two-block Gibbs sampling (<code>rLMMNormal_reg_known_vcov_two_block</code>) instead. Every other route (<code>dGamma()/dIndependent_Normal_Gamma</code> components, variance components estimated) only supports two-block Gibbs, so both values behave identically there. Ignored when <code>simulate = FALSE</code> .
<code>simulate</code>	Logical (default TRUE). When TRUE, draw posterior samples. When FALSE, return point estimates only via <code>rmerb_point</code> (no MCMC).

Details

The returned sample is `n` mutually independent draws — one per replicate chain, taken at that chain's final sweep, where the number of sweeps is calibrated in advance so the final sweep is

within `tv_tol` of the target posterior. It is therefore analyzed like an iid sample rather than like an MCMC trace.

This is the matrix-level interface. For a formula interface, use `lmerb()` in **lmebayes**. The Poisson/binomial/Gamma counterpart is [rglmerb](#), and the non-hierarchical analogue is [rlmb](#).

The model being estimated:

`rlmerb` estimates the two-stage Gaussian mixed model in a *centered* (Lindley–Smith) parameterization (Lindley and Smith 1972). For group $j = 1, \dots, J$:

$$y_j \mid \beta_j, \sigma_j^2 \sim N(D_j \beta_j, \sigma_j^2 I_{n_j}) \quad (\text{within group})$$

$$\beta_j \mid \gamma, \Psi \sim N(\mathcal{W}_j \gamma, \Psi) \quad (\text{across groups})$$

Read the second line as the model for how groups differ: each group's own coefficient vector β_j is drawn around a population expectation $\mathcal{W}_j \gamma$, with Ψ measuring how far groups scatter from it. The estimand is therefore the pair (γ, Ψ) at the population level together with all J group vectors β_j , and σ^2 for the residual scale.

What distinguishes this from the **lme4** presentation is that the *full* coefficient β_j appears in the likelihood, not a mean-zero deviation. Marginalizing recovers the familiar form with $X_j = D_j \mathcal{W}_j$, $Z_j = D_j$, and $b_j = \beta_j - \mathcal{W}_j \gamma$, so γ corresponds to **lme4**'s fixed effects and b_j to its random effects. The practical consequence is that `grouped` holds β_j directly (comparable to `lme4::coef()`), not b_j .

Ψ is diagonal, $\Psi = \text{diag}(\tau_1^2, \dots, \tau_{p_{re}}^2)$, because `model_setup` accepts only uncorrelated (`||`) random-effect terms and a single grouping factor. There is no random-effect correlation parameter to estimate.

What you supply:

Three arguments carry the statistical content; the rest are controls.

`design` A `model_setup` object, which fixes y , D_j , the grouping factor, and the level-2 design $\mathcal{W}_j$.

It also reports whether the data can identify the model at all; see `vignette("Chapter-B08", package = "lmebayesCore")`.

`pfamily_list` The prior on the population level, one component per random-effect coefficient.

This is where you declare whether the between-group variance τ_k^2 is **known** (`dNormal`, treating the supplied dispersion as fixed) or **estimated** (`dIndependent_Normal_Gamma`, placing a Gamma prior on the precision $1/\tau_k^2$).

`dispprior_list` The prior on the residual variance σ^2 : a fixed value (scalar, or one per group), a single `dGamma()` to estimate one pooled σ^2 , or a named list of `dGamma()` objects to estimate a separate σ_j^2 per group.

Both priors are normally produced from one `Prior_Setup_GLMM` object, via `pfamily_list()` and either `ps$group.dispersion` or `dGamma_list(ps)`.

Those two choices are also what determine which internal engine runs. You never name an engine; it follows from the priors. The statistically meaningful consequence is that when σ^2 is fixed and every component is `dNormal`, the posterior is exactly multivariate normal and can be sampled directly; every other combination requires iterative sampling. See `vignette("Chapter-B01", package = "lmebayesCore")` for the full mapping and `vignette("Chapter-B04", package = "lmebayesCore")` for the exact route.

What the returned draws are:

This is the part most likely to differ from other samplers, and it governs how the output should be analyzed.

rmerb runs n **independent replicate chains** and keeps **one draw from each** — the state after that chain's final sweep. It does not run one long chain and thin it. Two consequences:

1. **The n draws are mutually independent.** They are a genuine iid sample, not an autocorrelated series. Ordinary Monte Carlo standard errors apply directly; there is no effective sample size to compute, nothing to thin, and no trace plot or $\hat{R}$ to inspect on the returned sample.
2. **Each draw comes from a distribution deliberately made close to the target.** The sweep count `m_convergence` is not a burn-in guess. It is calibrated *before* sampling so that the distribution reached at the final sweep is within `tv_tol` of the exact posterior in total variation. `tv_tol` is a per-draw guarantee, and it is reported on the fit.

So n and `tv_tol` control different errors and can be set independently: n controls Monte Carlo noise, `tv_tol` controls the residual bias of each draw. To certify the whole n -draw sample jointly at level α , pass `tv_tol = alpha / n`; because cost grows only logarithmically in $1/\text{tv_tol}$, this typically adds a few sweeps.

In the exactly-Gaussian case (fixed σ^2 , all `dNormal`) the draws are exact rather than approximate: `m_convergence` is 1 and no approximation is involved.

How `m_convergence` is derived, and what to do when a warning reports strong cross-block coupling, is covered in `vignette("Chapter-B02", package = "lmebayesCore")` and `vignette("Chapter-B03", package = "lmebayesCore")`.

Point estimates without sampling:

`simulate = FALSE` returns only the posterior mean of (γ, β) at fixed variance-component plug-ins, computed by iterated conditional modes. No sweeps run and no RNG is consumed, so the result is deterministic and fast — useful when iterating on model specification. All draw and convergence slots are `NULL`. This same quantity is where the replicate chains start when `simulate = TRUE`, which is why it is always computed.

Value

An object of class `c("rmerb", "list")`. Each draw slot holds one row (or one value) per replicate chain, taken at that chain's final sweep; see the "What the returned draws are" section above.

Population level (γ , the analogue of **lme4** fixed effects):

`popef` Named list, one entry per random-effect coefficient. Entry k is an $n \times q_k$ matrix of draws, columns named by `colnames(design$W[[k]])`.

`popef.means`, `popef.mode` Named lists in the same shape: the mean of the draws, and the posterior mode used to start every chain. `popef.init` records the starting values.

Group level (β_j , one vector per group):

`groupef` Data frame in long format with $n \times J$ rows: a draw index, a column named for the grouping factor, then one column per random-effect coefficient. These are full coefficients, comparable to `lme4::coef()`, not mean-zero deviations.

`groupef.mode` $J \times p_{re}$ matrix of posterior modes, rows named by group level.

Variance components:

popecf.dispersion $n \times p_{re}$ matrix of between-group variance draws τ_k^2 . Constant down a column for a dNormal component (the value was treated as known); varying for an dIndependent_Normal_Gamma component. popecf.dispersion.mean is the column mean.

group.dispersion Residual variance σ^2 : a fixed scalar or per-group vector, or draws when a dGamma measurement prior was used. group.dispersion.mean is the corresponding mean.

Convergence reporting:

m_convergence Integer sweeps run per chain; 1 on the exact route.

convergence_info List with method ("exact_iid" when the draws are exact), lambda_star and eigenvalues (the contraction rate behind the sweep count), draw_engine, sim_method_used, and icm_info for the mode solve. convergence is the same list.

Inputs echoed back: design, family, n, call, pfamily_list, dispprior_list, prior.weights, offset, plus the packed prior and thin Prior summaries.

When simulate = FALSE, every draw and convergence slot is NULL and only the mode fields are populated.

See Also

[rglmerb](#), [rlmerb_point](#), [rLMMNormal_reg_known_vcov](#), [rLMMNormal_reg_estimated_vcov](#), [rLMMindepNormalGamma_reg_known_vcov](#), [rLMMindepNormalGamma_reg_estimated_vcov](#), [Prior_Setup_GLMM](#), [rlmb](#)

summary.bglmb

Summarize bglmb fits

Description

Summary method for row-block [glmb](#) fits (class "bglmb"). summary.bglmb applies [summary.glmb](#) to each block; print.summary.bglmb follows [summary.mlmb](#) / [summary.blmb](#) with per-block sections.

Usage

```
## S3 method for class 'bglmb'
summary(object, ...)

## S3 method for class 'summary.bglmb'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object	An object of class "bglmb" from glmbBlock .
x	An object of class "summary.bglmb".
digits	Number of significant digits for printing.
...	Passed to summary.glmb or print methods.

Value

summary.bglmb returns a named list of "summary.glmb" objects with class "summary.bglmb".

See Also

[glmbBlock](#), [lmbBlock](#), [summary.mlmb](#), [summary.blmb](#), [summary.glmb](#)

summary.blmb	<i>Summarize blmb fits</i>
--------------	----------------------------

Description

Summary method for row-block [lmb](#) fits (class "blmb").

Usage

```
## S3 method for class 'blmb'
summary(object, ...)

## S3 method for class 'summary.blmb'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

- object An object of class "blmb" from [lmbBlock](#).
- x An object of class "summary.blmb".
- digits Number of significant digits for printing.
- ... Passed to [summary.glmb](#) or print methods.

summary.lmerb	<i>Summarize Bayesian linear mixed model fits</i>
---------------	---

Description

Methods for [lmerb](#) fits. summary.lmerb builds Block~2 (level-2 fixed effect) tables per random-effects component, following the layout of [summary.glmb](#) and the multi-response structure of [summary.mlmb](#).

Usage

```
## S3 method for class 'lmerb'
summary(object, groups = NULL, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'glmerb'
summary(object, groups = NULL, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.lmerb'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

object	An object of class "lmerb" or "glmerb".
groups	Optional character vector of grouping levels for which to include a per-group Block~1 (random effects) detail table. When NULL (default), only an aggregate ranef_overview is returned.
digits	Number of significant digits for printing.
x	An object of class "summary.lmerb".
...	Ignored.

Value

summary.lmerb returns an object of class "summary.lmerb", a list with components call, formula, n, simulated, varcor, fixef_prior_overview (stacked prior and glmer/lmer reference across RE components), fixef_overview (Block~2 hyperparameters with posterior summaries and Pr(Prior_tail)), fixef_percentiles_overview (stacked distribution percentiles across RE components, when simulated), popef (per-RE-component tables; not printed, available on the returned object), ranef_overview, ranef_iters.mean (Block~1 envelope candidates per inner sweep, averaged over groups; printed separately from the overview table), any_non_normal, tau2_prior_overview (per-component τ_k^2 prior reference: Block~2 pfamily name (Prior), $1/E[1/\tau^2]$, $E[\tau^2]$, truncation window, $\sqrt{E[\tau^2]}$), and lmer/glmer MLE), sim_method_used ("DEFAULT" for the exact-iid engine, "TWO_BLOCK_GIBBS" for the two-block Gibbs engine, or NULL when simulated is FALSE; see [lmerb](#)'s sim_method), tau2_overview and tau2_percentiles_overview (posterior mode, mean, SD on the variance scale, Mean SD, and tau^2 quantiles when simulated), tau2_sd_percentiles_overview (2.5\sqrt{tau^2} draws vs lmer/glmer SD), a Residual row for observation-level σ^2 when dispersion_ranef is supplied (same columns as the τ_k^2 rows; not used for per-group gamma_list priors — see [summary_sigma2](#)), and optionally ranef_groups.

See Also

[lmerb](#), [glmerb](#), [print.lmerb](#), [summary_sigma2](#), [summary.glmb](#), [summary.mlmb](#)

summary.ranef.lmerb	<i>Summarize ranef.lmerb objects</i>
---------------------	--------------------------------------

Description

Summarize ranef.lmerb objects

Usage

```
## S3 method for class 'ranef.lmerb'  
summary(object, groups = NULL, digits = 4L, ...)
```

Arguments

- object An object of class "ranef.lmerb".
- groups Optional character vector of grouping levels to highlight.
- digits Number of digits for printing.
- ... Ignored.

Value

A data.frame (invisibly) with long-format random effects.

summary_sigma2	<i>Summarize per-group measurement dispersion (sigma^2)</i>
----------------	---

Description

Per-group observation-level σ^2 summaries when dispersion_ranef is per-group: either a named list of [dGamma\(\)](#) priors (dispersion_mode == "gamma_list") or a named numeric vector of fixed, known values (dispersion_mode == "fixed_vector").

For "gamma_list", the layout mirrors the Block~2 τ_k^2 tables in [summary.lmerb](#): a prior reference table (with a common lmer/glmer residual column), an optional posterior overview from stored draws (including per-group Cand/draw, envelope candidates per inner sweep), and per-group distribution percentiles of the σ^2 draws (1\ 99\ the prior table shows whether the simulation tends to pile up against the truncation bounds. The prior table also reports each group's implied effective prior sample size (n_prior, back-computed from shape_ING under the Prior_Setup() default k = 1 calibration) and observed data sample size (n_data); the overview table reports their sum (n_total).

For "fixed_vector", there is no prior to calibrate and nothing is sampled (each group's σ_j^2 is known and constant throughout), so only a single table of the fixed per-group values next to the lmer/glmer residual variance is returned; overview and percentiles are NULL.

Usage

```
summary_sigma2(object, ...)

## S3 method for class 'lmerb'
summary_sigma2(
  object,
  type = c("both", "prior", "overview"),
  digits = max(3L, getOption("digits") - 3L),
  ...
)

## S3 method for class 'glmerb'
summary_sigma2(
  object,
  type = c("both", "prior", "overview"),
  digits = max(3L, getOption("digits") - 3L),
  ...
)

## S3 method for class 'summary.sigma2.lmerb'
print(x, digits = NULL, ...)

## S3 method for class 'summary.sigma2.lmerb'
as.data.frame(x, row.names = NULL, ...)
```

Arguments

object	An lmerb or glmerb fit.
type	"both" (default), "prior", or "overview".
digits	Number of significant digits for printing.
x	An object of class "summary.sigma2.lmerb".
row.names	Ignored.
...	Ignored.

Value

An object of class "summary.sigma2.lmerb" with components prior, overview, percentiles (overview and percentiles are always NULL for "fixed_vector"), group_name, n_groups, simulated, mode, and mer_label. When window_diagnostics were attached by [dGamma_list\(\)](#), the prior table includes blup_infl, R_lo, R_hi, and asymmetric_window.

See Also

[summary.lmerb](#), [lmerb](#).

Index

- * **Bayesian Binomial Regression**
 - AMI, [5](#)
 - Cleveland, [10](#)
- * **Bayesian Gamma Regression**
 - carinsca, [9](#)
- * **Bayesian Poisson Regression**
 - carinsca, [9](#)
- * **Bayesian linear regression**
 - Boston_centered, [8](#)
- * **Bike sharing**
 - BikeSharing, [6](#)
- * **Count regression**
 - BikeSharing, [6](#)
- * **datasets**
 - AMI, [5](#)
 - BikeSharing, [6](#)
 - Boston_centered, [8](#)
 - carinsca, [9](#)
 - Cleveland, [10](#)
- * **modelfuncs**
 - glmbBlock, [23](#)
 - lmbBlock, [37](#)
- AMI, [5](#)
- anova.glmb, [15](#)
- as.data.frame, [17](#), [24](#), [32](#), [38](#), [62](#), [78](#)
- as.data.frame.ranef.lmerb, [6](#)
- as.data.frame.summary.sigma2.lmerb
(summary_sigma2), [91](#)
- BikeSharing, [6](#)
- binomial, [20](#)
- Boston, [8](#)
- Boston_centered, [8](#)
- build_mu_all, [45](#), [46](#), [76](#)
- carinsca, [9](#)
- check_identifiability, [49–51](#)
- Cleveland, [10](#)
- coef.glmerb (coef.lmerb), [11](#)
- coef.lmerb, [11](#), [15](#), [80](#)
- coef.merMod, [11](#), [12](#)
- coefficients, [19](#), [34](#)
- compute_gaussian_prior, [54](#), [63](#), [66](#)
- dBeta, [18](#)
- demo, [30](#)
- dGamma, [12](#), [18](#), [41](#), [53](#), [54](#), [57](#), [67](#), [72](#), [76](#), [85](#),
[86](#), [91](#)
- dGamma (pfamily), [52](#)
- dGamma_list, [12](#), [41](#), [57](#), [69](#), [76](#), [86](#), [92](#)
- diagnose_glmbayes, [31](#)
- dIndependent_Normal_Gamma, [53](#), [54](#), [57](#), [67](#)
- dIndependent_Normal_Gamma (pfamily), [52](#)
- directional_tail, [14](#)
- dNormal, [57](#), [67](#), [73](#)
- dNormal (pfamily), [52](#)
- dNormal_Gamma, [53](#), [54](#), [57](#), [67](#)
- dNormal_Gamma (pfamily), [52](#)
- dummy.coef.glmb, [20](#), [36](#)
- EnvelopeBuild, [4](#), [20](#), [35](#), [56](#)
- extract_re_hyper_matrices, [48–51](#)
- extractAIC, [18](#), [19](#), [34](#)
- extractAIC.glmb, [20](#), [36](#)
- family, [16](#), [19](#), [20](#), [23](#), [24](#), [26](#), [50](#), [55](#), [69](#), [81](#)
- fitted.values, [18](#), [19](#), [34](#)
- fixef, [15](#)
- fixef.glmerb (fixef.lmerb), [15](#)
- fixef.lmerb, [15](#), [80](#)
- formula, [16](#), [20](#), [24](#), [32](#), [38](#), [62](#), [78](#)
- glm, [18–20](#), [24](#), [32](#), [33](#), [35](#), [38](#), [48](#), [64](#)
- glm.control, [17](#)
- glmb, [4](#), [16](#), [19](#), [23–25](#), [28](#), [30](#), [34–36](#), [44](#), [46](#),
[88](#)
- glmbBlock, [3](#), [4](#), [23](#), [39](#), [88](#), [89](#)
- glmer, [27–29](#), [47](#), [48](#), [50](#), [69](#)
- glmerb, [25](#), [46](#), [80](#), [90](#)

- glmerb_posterior_mode, 30
- glmerControl, 28, 48, 69
- glmmTMB, 29, 44, 45, 50
- has_opengl, 31, 31
- lFormula, 48, 69
- lm, 19, 20, 33–35, 39
- lmb, 4, 19, 20, 27, 31, 34, 37, 39, 42–44, 46, 89
- lmbBlock, 3, 4, 23, 25, 37, 89
- lmebayes (lmebayes-package), 3
- lmebayes-package, 3
- lmer, 28, 42–44, 47–50, 69
- lmerb, 4, 27–30, 40, 80, 89, 90, 92
- lmerb_posterior_covariance, 61
- lmerb_posterior_mean, 45, 46, 59
- lmerControl, 28, 43, 48, 69
- load_kernel_source, 31
- model.extract, 17
- model.offset, 63, 78
- model_setup, 3, 4, 26, 28, 29, 41, 43, 45, 46, 47, 68, 69, 73, 74, 76, 81, 82, 84–86
- multi_prior_setup, 67, 80
- na.action, 63, 78
- na.fail, 17, 24, 32, 38
- normalize_group, 24, 38, 78–80
- offset, 17, 63, 78
- options, 17, 24, 32, 38, 63, 78
- packageStartupMessage, 4
- pfamily, 17, 20, 26, 32, 33, 36, 38, 41, 52, 56, 57, 67, 79
- pfamily_list, 12, 26, 41, 56, 74–76, 80, 86
- plot_mean_convergence, 58, 60, 61
- plot_mean_convergence.default, 58
- plot_var_convergence, 58, 59, 59
- plot_var_convergence.default, 60
- predict, 18
- predict.glmb, 20, 36
- print.bglmb (glmbBlock), 23
- print.blmb (lmbBlock), 37
- print.glmb, 18
- print.glmerb (glmerb), 25
- print.lmerb, 90
- print.lmerb (lmerb), 40
- print.ranef.lmerb, 61
- print.summary.bglmb (summary.bglmb), 88
- print.summary.blmb (summary.blmb), 89
- print.summary.lmerb (summary.lmerb), 89
- print.summary.sigma2.lmerb (summary_sigma2), 91
- print.two_block_sweep_history, 58, 60
- Prior_Check, 20, 36, 56, 67
- Prior_Setup, 18, 20, 33, 36, 53–56, 62, 69–71, 76, 77, 79, 80
- Prior_Setup_GLMM, 12, 26, 29, 41, 43, 46, 48–50, 57, 67, 79, 80, 84, 86, 88
- Prior_SetupGroup, 57, 77
- prior_simfuncs, 54, 56
- ranef, 6, 80
- ranef.glmerb (ranef.lmerb), 80
- ranef.lmerb, 12, 80
- rBeta_reg, 55
- residuals, 18, 19, 34
- residuals.glmb, 20
- rGamma_Conjugate_reg, 55, 56
- rGamma_prior, 56
- rGamma_reg, 54, 56
- rglmb, 4, 19, 34, 35, 52, 53, 56, 67
- rglmerb, 28, 81, 86, 88
- rglmerb_point, 82, 84
- rGLMM_reg, 81, 84
- rGLMM_sweep, 60
- rindepNormalGamma_reg, 26, 41, 56, 67
- rlmb, 4, 19, 34, 35, 52, 53, 56, 67, 86, 88
- rlmerb, 27, 42, 82, 83, 84, 84
- rlmerb_point, 85, 88
- rlMM_reg, 12
- rlMMindepNormalGamma_reg_estimated_vcov, 84, 88
- rlMMindepNormalGamma_reg_known_vcov, 84, 88
- rlMMNormal_reg, 81, 85
- rlMMNormal_reg_estimated_vcov, 84, 88
- rlMMNormal_reg_known_vcov, 42, 84, 88
- rlMMNormal_reg_known_vcov_iid, 85
- rlMMNormal_reg_known_vcov_two_bg, 85
- rNormal_Gamma_prior, 56
- rNormal_prior, 56
- rNormal_reg, 54, 56
- rNormal_reg_group, 46, 77, 80
- rNormalGamma_reg, 56
- rNormalGLM_reg_group, 23, 37, 77, 80
- simfuncs, 67

simfunction, [3](#), [4](#)
simulate.glmb, [20](#), [36](#)
summary.bglmb, [23](#), [88](#)
summary.blmb, [88](#), [89](#), [89](#)
summary.glmb, [15](#), [18–20](#), [34](#), [36](#), [88–90](#)
summary.glmerb(summary.lmerb), [89](#)
summary.lmerb, [89](#), [91](#), [92](#)
summary.mlmb, [88–90](#)
summary.ranef.lmerb, [91](#)
summary_sigma2, [90](#), [91](#)

terms, [19](#), [35](#), [50](#)
two_block_optimize_pilot_cost, [27](#)
two_block_rate_from_pfamily_list, [27](#)
two_block_rNormal_reg, [46](#), [69](#)

VarCorr, [50](#)