Package: lmebayes (via r-universe)

June 21, 2026

Type Package

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

Version 0.1.0

Date 2026-06-01

Description Provides Bayesian linear and generalized linear mixed-effects model fitting with near-independent posterior samples. The main functions mirror lme4's 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 'glmbayesCore' package.

License GPL (>= 2)

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

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

Imports glmbayesCore, glmbayes, stats, Matrix, lme4, reformulas, 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, LearnBayes, testthat (>= 3.0.0), spelling

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 ocl-icd-opencl-dev

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

Date/Publication 2026-06-21 14:26:43 UTC

RemoteUrl https://github.com/knygren/lmebayes

RemoteRef HEAD

RemoteSha a000d5fc7e386afd7ea5cde0d75ba82cd50a791b

Contents

lmebayes-package *lmebayes: Bayesian Linear Mixed-Effects Models via Two-Block Gibbs Sampling*

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 **glmbayesCore**.

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

OpenCL startup checks

In interactive sessions, attaching the package with `library(lmebayes)` may emit a short `packageStartupMessage` when `has_opencl()` 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_opencl()` then reports whether that build succeeded. Set `options(glmbayes.quiet_opencl_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())

ps <- glmbayesCore::Prior_Setup(counts ~ outcome + treatment, family = poisson())

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

AMI *Amitriptyline overdose data*

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
###############################################################################
```

`BikeSharing` *Bike Sharing Dataset (Processed)*

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
#############################################################################
```

`directional_tail` *Directional Tail Diagnostic*

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

`fit`	A fitted model object of class 'glmb' or 'lmb'
`mu0`	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:

`mahalanobis_shift`	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‖).
`p_directional`	Directional tail probability (proportion of draws in the direction of disagreement)
`delta`	Mean shift in whitened space
`draws`	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`

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 glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)

pfamily	a description of the prior distribution and associated constants to be used in the model. This should be a pfamily function (see pfamily for details of pfamily functions).
n	number of draws to generate. If length(n) > 1, 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 environment(formula), typically the environment from which glm is called.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
use_parallel	Logical. Whether to use parallel processing during simulation.
use_opencl	Logical. Whether to use OpenCL acceleration during Envelope construction.
verbose	Logical. Whether to print progress messages.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
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 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 model.offset at model.extract.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} can be useful.
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.
start	starting values for the parameters in the linear predictor.
etastart	starting values for the linear predictor.
mustart	starting values for the vector of means.
control	a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control.
model	a logical value indicating whether *model frame* should be included as a component of the returned value.
method	the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS): the alternative "model.frame" 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.

x, y	For glm: 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 glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
...	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

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_opencl = 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:

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

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 block_rNormalGLM 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_opencl = 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 nrow(data) (after model.frame), a column name in data, l2_blocks counts, or a list of row index vectors (see normalize_block).
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function. For glm.fit only the third option is supported. (See family for details of family functions.)
pfamily	Recycled to all blocks, or use pfamily_list of length k.
pfamily_list	Optional list of pfamily objects, one per block.
n	number of draws to generate. If length(n) > 1, 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 environment(formula), typically the environment from which lm 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 NULL or a numeric vector. If non-NULL, weighted least squares is used with weights weights (that is, minimizing sum(w*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 na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} 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 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.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
model, x, y	For glmbBlock, logicals passed to each block glmb fit (see glm). For print, x is the "bglmb" 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 EnvelopeOpt for grid construction. Defaults to match n. Larger values encourage tighter envelopes.
use_parallel	Logical. Whether to use parallel processing during simulation.
use_opencl	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 "bglmb" (list of "glmb" fits from **glmbayes**).

Functions

- glmbBlock(): glmb fit per row block.

See Also

glmbayes Modeling Functions lmbBlock()

---

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

Description

Bayesian generalized linear mixed-effects model fit

Usage

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

```
  offset,
  contrasts = NULL,
  devFunOnly = FALSE,
  fixef = NULL,
  progbar = FALSE,
  ...
)

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

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 gaussian().
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 (mu, Sigma) 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 (rindepNormalGamma_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 [disp_lower, 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 $n_{\text{prior}} \leq J$ (pwt_dispersion ≤ 0.5). Typically built with pfamily_list from a Prior_Setup_lmebayes object.
dispersion_ranef	Observation-level measurement dispersion, treated as known during sampling. Required positive scalar for families with a dispersion parameter (e.g. gaussian()); must be NULL (default) for poisson() and binomial(). Typically Prior_Setup_lmebayes(...)$dispe
n	Number of iid draws per group (default 1000L, as in lmb).
gap_tol	Tolerated mode–mean gap in units of posterior standard deviations (default 0.0196). Applies only to non-Gaussian families. The number of pilot chains is derived from this tolerance as n_pilot = ceiling((qnorm(0.975) / gap_tol)^2), which ensures that a gap larger than gap_tol posterior SDs is detected with 95\ gap_tol = 0.0196 = 1.96 / 100 gives n_pilot = 10000. The pilot stage runs n_pilot independent chains from the ICM mode, each returning one stored draw after m_convergence_pilot inner sweeps, and takes the column-means as coef.pilot.mean – the starting point for the main run. Set gap_tol = NULL to skip the pilot entirely and start the main run from the ICM mode (as in the Gaussian case). Ignored for family = gaussian(), where mode equals mean exactly.

`mode_gap_max` Maximum per-coordinate mode–mean gap (in posterior standard deviation units) that `m_convergence_pilot` is calibrated to cover (default `1.0`). Applies only to non-Gaussian families when `gap_tol` is not `NULL` and `m_convergence_pilot` is `NULL`. The pilot chains start at the ICM mode, which is at Mahalanobis distance $D_{\max} = \sqrt{p} \times$ `mode_gap_max` from the posterior mean (assuming `mode_gap_max` SDs per coordinate across p fixed-effect dimensions). The number of pilot sweeps is the smallest l satisfying $\mathrm{erf}_1(0.5\,\lambda^{*l}\,D_{\max}/\sqrt{2}) \leq$ `tv_tol` (Nygren 2020, Theorem 3 mean-shift term), floored at `m_min`. Set `mode_gap_max = NULL` to fall back to `m_convergence` (pre-v0.2 behaviour). Ignored for `family = gaussian()`.

`tv_tol` Total variation tolerance per stored draw, in (0, 1) (default `0.01`). For `family = gaussian()` the joint posterior is exactly multivariate normal and the number of inner Gibbs sweeps per stored draw is calibrated exactly as in `lmerb` (Nygren 2020, Theorem 3). For non-Gaussian families the same calibration is applied to the *local-Gaussian approximation of the posterior at its mode*: per-observation likelihood precisions are evaluated at the ICM posterior mode (`two_block_mode_weights`) and fed to `two_block_rate_v2`. The derived sweep count is then the *minimum* number of iterations required to converge to that hypothetical multivariate normal approximation – a lower bound for the true (non-normal) posterior, not a guarantee.

`m_convergence` Optional integer override for the number of inner Gibbs sweeps per stored draw. When `NULL` (default) the `tv_tol`-derived value is used. A supplied value acts as a requested sweep count but is never allowed below the derived minimum: `max(m_convergence, m_min)` is used, with a warning if the value had to be raised. Typical use is to pick a *larger* number for non-Gaussian families (e.g. double the derived lower bound).

`m_convergence_pilot` Optional integer override for the number of inner Gibbs sweeps used in each independent pilot chain. Applies only when `gap_tol` is not `NULL` and `family` is non-Gaussian. When `NULL` (default), the sweep count is derived automatically from `mode_gap_max`, `tv_tol`, and `rate$lambda_star` via Theorem 3 (see `mode_gap_max`). A supplied value is used as-is (no floor is applied beyond what the user intends) and overrides the `mode_gap_max` derivation.

`simulate` Logical (default `TRUE`). When `TRUE` the two-block Gibbs sampler is run for `n` iterations and posterior draws are stored. When `FALSE` only the ICM algorithm is run: the exact posterior means (`fixef.mode`, `ranef.mode`) are computed and returned immediately without any sampling. Simulation-only fields (`coefficients`, `fixef.means`, `fixef`) are `NULL` when `simulate = FALSE`.

`REML` Logical; passed to `model_setup`.

`control` Optional `glmerControl` settings passed to the reference `glmer` fit. Defaults to `NULL` (lme4 defaults). When `family = gaussian()`, lme4's `glmer` shortcut to `lmer` does not accept an explicit `glmerControl`; leave `control = NULL` or pass `lmerControl`.

`start` Optional starting values; passed to `model_setup`.

`verbose` Verbosity flag; passed to `model_setup`.

`subset` Optional subset; passed to `model_setup`.

weights	Optional weights; passed to model_setup.
na.action	Missing-data handler; passed to model_setup.
offset	Optional offset; passed to model_setup.
contrasts	Optional contrasts; passed to model_setup.
devFunOnly	If TRUE, return deviance function only; passed to model_setup.
fixef	Optional named list of hyper-parameter vectors (Block 2 state). When NULL (default), iter-0 means are taken from the pfamily_list prior means.
progbar	Logical. When TRUE, show text progress bars during pilot and main replicate sampling inside rglmerb. Default FALSE.
...	Reserved for future use.
x	Object of class "glmerb".
digits	Number of significant digits to use when printing.

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.

Currently a copy of lmerb with an additional family argument. When family = gaussian(), behaviour matches lmerb except that the embedded reference fit is from glmer rather than lmer. Non-Gaussian families use block_rNormalGLM for Block~1 Gibbs updates.

Value

Object of class "glmerb": same fixef.* structure as "lmerb", with additional family, glmer (reference glmer fit), fixef.init (main-chain start from pilot colMeans when a pilot runs; NULL for Gaussian or when n_pilot = NULL), pilot_chisq (Hotelling chi-squared test of pilot mean vs ICM mode), gap_tol, and mode_gap_max.

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_lmebayes(form, data = dat, family = poisson(), 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_opencl *OpenCL compile-time status for* **lmebayes**

Description

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

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

Usage

```
has_opencl()
```

Value

Logical scalar.

See Also

opencltools, **glmbayes**

lmb *Fitting Bayesian Linear Models*

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_opencl = 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. cbind(y1, y2) ~ x) 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 length(n) > 1, 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 environment(formula), typically the environment from which lm 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 NULL or a numeric vector. If non-NULL, weighted least squares is used with weights weights (that is, minimizing sum(w*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 na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} can be useful.
method	the method to be used in fitting the classical model during a call to glm. The default method glm.fit uses iteratively reweighted least squares (IWLS): the alternative "model.frame" 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	logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.
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_opencl	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).

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 glmb.

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:

lm	an object of class "lm" containing the output from a call to the function lm
coefficients	a matrix of dimension n by length(mu) with one sample in each row

coef.means	a vector of length(mu) with the estimated posterior mean coefficients
coef.mode	a vector of length(mu) with the estimated posterior mode coefficients
dispersion	Either a constant provided as part of the call, or a vector of length n with one sample in each row.
Prior	A list with the priors specified for the model in question. Items in list may vary based on the type of prior
residuals	a matrix of dimension n by length(y) with one sample for the deviance residuals in each row
fitted.values	a matrix of dimension n by length(y) with one sample for the fitted values in each row
linear.predictors	an n by length(y) matrix with one sample for the linear fit on the link scale in each row
deviance	an n by 1 matrix with one sample for the deviance in each row
pD	An Estimate for the effective number of parameters
Dbar	Expected value for minus twice the log-likelihood function
Dthetabar	Value of minus twice the log-likelihood function evaluated at the mean value for the coefficients
DIC	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 * 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 classed 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()`, `rglmb()`, `rlmb()`

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 block_rNormalGLM (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_opencl = FALSE,
  verbose = FALSE,
  ...
)

## S3 method for class 'blmb'
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 nrow(data) (after model.frame), a column name in data, l2_blocks counts, or a list of row index vectors (see normalize_block).
pfamily	A single pfamily recycled to every block, or use pfamily_list of length k (number of blocks).
pfamily_list	Optional list of pfamily objects, one per block.

n	number of draws to generate. If length(n) > 1, 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 environment(formula), typically the environment from which lm 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 NULL or a numeric vector. If non-NULL, weighted least squares is used with weights weights (that is, minimizing sum(w*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 na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is stats{na.omit}. Another possible value is NULL, no action. Value stats{na.exclude} can be useful.
method	the method to be used in fitting the classical model during a call to glm. The default method glm.fit uses iteratively reweighted least squares (IWLS): the alternative "model.frame" 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_opencl	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_SetupBlock(
  Sepal.Length ~ Sepal.Width + Petal.Length,
  block = "Species",
  data = iris,
  family = gaussian()
)

pfamily_list <- lapply(ps_block, function(ps) {
  glmbayesCore::dNormal_Gamma(
    mu = ps$mu, Sigma_0 = ps$Sigma_0, shape = ps$shape, rate = ps$rate
  )
})

out_blmb <- lmbBlock(
  Sepal.Length ~ Sepal.Width + Petal.Length,
  block = "Species",
  pfamily_list = pfamily_list,
  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,
  n = 1000L,
  tv_tol = 0.01,
  m_convergence = NULL,
  simulate = TRUE,
  REML = TRUE,
  control = lme4::lmerControl(),
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  devFunOnly = FALSE,
  fixef = NULL,
  seed = NULL,
  ...
)

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

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 (mu, Sigma) 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 (rindepNormalGamma_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 [disp_lower, 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 $n_{\text{prior}} \leq J$ (pwt_dispersion ≤ 0.5). Typically built with pfamily_list from a Prior_Setup_lmebayes object.

dispersion_ranef
: Required positive scalar: the observation-level measurement dispersion σ^2, treated as known during sampling. Typically `Prior_Setup_lmebayes(...)$dispersion_ranef`. (A prior specification for this parameter may be supported in the future.)

n
: Number of iid draws per group (default `1000L`, as in `lmb`).

tv_tol
: 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 `tv_tol` of the exact joint posterior in total variation (Nygren 2020, Theorem 3; see Details). To certify the whole `n`-draw sample at level α pass `tv_tol = alpha / n`; the cost grows only logarithmically in `1/tv_tol`.

m_convergence
: Optional integer override for the number of inner Gibbs sweeps per stored draw. When `NULL` (default) the `tv_tol`-derived value is used. A supplied value is floored at the derived minimum: `max(m_convergence, m_min)` is used, with a warning if the value had to be raised.

simulate
: Logical (default `TRUE`). When `TRUE` the two-block Gibbs sampler is run for `n` iterations and posterior draws are stored. When `FALSE` only the ICM algorithm is run: the exact posterior means (`fixef.mode`, `ranef.mode`) are computed and returned immediately without any sampling. Simulation-only fields (`coefficients`, `fixef.means`, `fixef`) are `NULL` when `simulate = FALSE`.

REML
: Logical; passed to `model_setup`.

control
: `lmerControl` settings; passed to `model_setup`.

start
: Optional starting values; passed to `model_setup`.

verbose
: Verbosity flag; passed to `model_setup`.

subset
: Optional subset; passed to `model_setup`.

weights
: Optional weights; passed to `model_setup`.

na.action
: Missing-data handler; passed to `model_setup`.

offset
: Optional offset; passed to `model_setup`.

contrasts
: Optional contrasts; passed to `model_setup`.

devFunOnly
: If `TRUE`, return deviance function only; passed to `model_setup`.

fixef
: Optional named list of hyper-parameter vectors (Block 2 state). When `NULL` (default), iter-0 means are taken from the `pfamily_list` prior means.

seed
: Optional; sets the RNG seed before sampling.

...
: Reserved for future use.

x
: Object of class `"lmerb"`.

digits
: Number of significant digits to use when printing.

Details

Entry point for **lmebayes** models with an `lmer`-like interface, analogous to `lmb` and `glmb` for fixed-effects models.

Calls `model_setup` on `formula` and `data` for design matrices (`y`, `Z`, `groups`, `X_hyper`, etc.) and embeds the resulting `lmer` fit as `lmer`. Priors are supplied as a named list of `pfamily` objects

(pfamily_list, the Block~2 hyperpriors – one per random-effect coefficient) plus the observation-level measurement dispersion (dispersion_ranef). Both are typically built from Prior_Setup_lmebayes: pfamily_list = pfamily_list(ps) and dispersion_ranef = ps$dispersion_ranef. The Block~1 random-effect covariance is reconstructed from the Block~2 pfamily dispersions (Sigma_ranef = diag(tau^2_k)); lmerb does not call Prior_Setup_lmebayes internally.

Runs a two-block Gibbs sampler for n iterations. Block 1 draws group-level random effects b_j given the current hyper means; Block 2 updates the hyper means (level-2 fixed effects γ_k) given the current b_j draw, using multi_rNormal_reg with the hyper design matrices from design$X_hyper.

Exact posterior and convergence characterisation. When variance components are treated as fixed at their lmer estimates (as done here), the joint posterior over the random-effect coefficients and the level-2 fixed effects is *exactly* multivariate normal. In this setting the convergence of the two-block Gibbs sampler is fully characterised: Corollary 1 of Nygren (2020) shows that the total variation (TV) distance between the l-step kernel and the target density is bounded above by a geometrically decreasing function of l, with contraction rate $\lambda^* \in [0, 1)$, the maximal eigenvalue of the matrix

$$A \;=\; P_{11}^{-1/2}\, P_{12}\, P_{22}^{-1}\, P_{21}\, P_{11}^{-1/2}$$

where P_{11}, P_{22}, and P_{12} are the corresponding blocks of the joint precision matrix. Because the bound is explicit and computable, the required number of iterations to reach a pre-specified TV tolerance ε can be derived analytically once λ^* is known.

TV-calibrated m_convergence**.** The number of inner Gibbs sweeps per stored draw (m_convergence) is derived from tv_tol: lmerb computes the Remark 8 eigenvalue spectrum with two_block_rate_v2 and inverts the exact Theorem 3 bound with two_block_l_for_tv. Because every replicate chain is started at the exact joint posterior mean (computed by ICM via lmerb_posterior_mean), the mean term of the bound vanishes and only the variance-convergence sum remains. One extra sweep is added because the bound applies to the block updated second in each sweep (the level-2 fixed effects γ); the stored random-effect draw lags by a half-step. Each stored draw is therefore guaranteed to be within tv_tol of the exact joint posterior in total variation.

Value

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

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.

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.

fixef.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).

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

fixef.means Named list of posterior mean vectors computed as colMeans(fixef[[k]]) — the MCMC estimate of the level-2 fixed effects. NULL when simulate = FALSE.

`fixef` Named list of $n \times q_k$ matrices of Block 2 draws, one per RE component. `NULL` when `simulate = FALSE`.

`coefficients` `data.frame` with `n * J` rows: `draw`, the grouping-factor column, and one column per RE variable. Average over `draw` within each group for posterior means (see Examples). `NULL` when `simulate = FALSE`.

`fixef.dispersion` $n \times p_{\mathrm{re}}$ matrix of the Block~2 dispersion (τ_k^2) at each stored draw: sampled values for `dIndependent_Normal_Gamma` components, constant columns (the fixed `dispersion`) for `dNormal` components. `NULL` when `simulate = FALSE`.

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

`fixef.iters` $n \times p_{\mathrm{re}}$ 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_converg` 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 x J` of Block 1 prior means at the final Gibbs state (from `build_mu_all`).

`convergence` List describing the sweep-count calibration: `method` (`"exact"`, 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`.

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_lmebayes`, `model_setup`, `build_mu_all`, `two_block_rNormal_reg_v2`, `lmerb_posterior_mean`, `block_rNormalReg`, `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_lmebayes(form, data = dat, pwt = 0.01)

  fit <- lmerb(
    form,
    data             = dat,
    pfamily_list     = pfamily_list(ps),
    dispersion_ranef = ps$dispersion_ranef,
    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` *Bayesian mixed model setup (single-factor* `lmer`/`glmer` *gate)*

Description

Wrapper around `lmer` or `glmer` for models with exactly one grouping factor. Design matrices come from `formula` (including cross-level RE moderation terms). Variance components use `vcov_formula` (defaults to `lmerb_default_vcov_formula`): level-2 fixed only, same `||` random structure as `formula`, without cross-level fixed interactions (so RE moderation is not double-coded).

Usage

```
model_setup(
  formula,
  data = NULL,
  vcov_formula = NULL,
  family = gaussian(),
  REML = TRUE,
```

```
  control = NULL,
  start = NULL,
  verbose = 0L,
  subset,
  weights,
  na.action,
  offset,
  contrasts = NULL,
  devFunOnly = FALSE,
  fit_mer = TRUE,
  ...
)
```

Arguments

formula	Mixed-model formula for design extraction and the reference lmer/glmer fit (fixed effects / hyper calibration).
data	Optional data frame.
vcov_formula	Optional formula for the reference fit used to extract vcov_re. Defaults to lmerb_default_vcov_formula(formula).
family	A family object. Defaults to gaussian(), in which case lmer is used. Non-Gaussian families use glmer.
REML	Logical; passed to lmer when family = gaussian().
control	lmerControl when family = gaussian(), otherwise glmerControl; passed through to the reference fit when fit_mer = TRUE.
start	Optional starting values for the inner optimization.
verbose	Passed to lmer.
subset, weights, na.action, offset, contrasts	Passed to lmer.
devFunOnly	If TRUE, return the deviance function only (Gaussian lmer fits only).
fit_mer	If TRUE (default), fit reference lmer/glmer models and extract variance components. If FALSE, return design matrices and rank diagnostics only (used by glmerb).
...	Passed to design extraction and, when fit_mer = TRUE, to the reference lmer/glmer fit.

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`, `model_setup` performs a two-step rank check that assesses whether the model is empirically identified at both the within-group and across-group levels:

1. *Level 1 (within-group):* For each group j, the within-group random-effects design submatrix $\mathbf{Z}_j$ is checked for full column rank (`re_rank`). A rank-deficient group has too few distinct observations to estimate all random slopes independently; its BLUPs are identified through the prior rather than the data alone. Such groups are flagged in `re_rank` but are retained in the `lmer` fit; `Prior_Setup_lmebayes` excludes them when calibrating priors.
2. *Level 2 (across-group):* Restricting to the full-rank groups from Step 1, each hyper-design matrix `X_hyper[[k]]` is checked for full column rank (`hyper_rank`). Rank deficiency at this level means the level-2 hyperparameters $\boldsymbol{\mu}_k$—the prior means for random-effect coefficient k across groups—are not identified by the data, even as the number of full-rank groups grows.

The scalar `rank_ok` is `TRUE` only when every `X_hyper[[k]]` is full-rank after Step 2. This is a necessary condition for `Prior_Setup_lmebayes` to derive default priors automatically; models with `rank_ok = FALSE` require user-supplied hyperpriors.

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

Value

Object of class `"model_setup"`: `y`, `Z`, `groups`, `X_hyper`, `formula`, `family`, `vcov_formula`, `lmer_fit` / `glmer_fit` (full formula), matching `lmer_vcov_fit` / `glmer_vcov_fit`, `varcorr`, `vcov_re`, `residual_var`, and `re_rank` (named logical vector: `TRUE` if `Z_j` is full column rank for that group).

See Also

`extract_re_hyper_matrices`, `lmerb_default_vcov_formula`, `extract_lmer_variance_components`

Examples

```
## model_setup() on bayesrules::big_word_club
##
## Same model as the full lmerb demo, demo/Ex_12_lmerb_BigWordClub.R (see
## also Prior_Setup_lmebayes / lmerb development scripts in data-raw/).
##
## Level 1 (students):
##   y ~ b0[j] + b_ppvt[j]*distracted_ppvt + b_a1[j]*distracted_a1
##
## Level 2 (schools):
##   b0[j]      ~ private_school + title1 + free_reduced_lunch + u0[j]
```

```
##   b_ppvt[j]  ~ 1 + u_ppvt[j]
##   b_a1[j]    ~ 1 + free_reduced_lunch + u_a1[j]
##                  (cross-level: free_reduced_lunch:distracted_a1 in formula)
##
## Each random slope has a matching fixed main effect (required for
## Prior_Setup_lmebayes() default calibration).

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_a1 + distracted_ppvt +
  free_reduced_lunch:distracted_a1 +
  (1 + distracted_ppvt + distracted_a1 || school_id)

ctrl_bobyqa <- lme4::lmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 2e5))

## ------------------------------------------------------------------------
## 1. lmer fit: raw output
## ------------------------------------------------------------------------
cat("--- lmer fit ---\n")
fit <- lme4::lmer(form_lmer, data = dat, control = ctrl_bobyqa)
print(summary(fit))

cat("\n--- fixef(fit): population-level (gamma) estimates ---\n")
print(lme4::fixef(fit))

cat("\n--- coef(fit): per-group coefficients (fixef + ranef) ---\n")
print(coef(fit))

## ------------------------------------------------------------------------
## 2. model_setup: structured view of the same model
## ------------------------------------------------------------------------
design <- model_setup(form_lmer, data = dat, control = ctrl_bobyqa)
print(design)

## ------------------------------------------------------------------------
## 3. Random effects b[j]: first 10 schools
## ------------------------------------------------------------------------
cat("--- Random effects b[j]: first 10", design$group_name, "---\n")
re_df <- as.data.frame(lme4::ranef(design$lmer_fit)[[design$group_name]])
```

```
print(utils::head(re_df, 10))

## ------------------------------------------------------------------------
## 4. Gamma estimates organised to match the Random Effects Model above
##
##    Mapping (intercept RE): X_hyper columns map directly to fixef() names.
##    Mapping (slope RE, hyper ~ 1): (Intercept) column -> fixef[slope_name].
## ------------------------------------------------------------------------
cat("\n--- Random effects model (gamma estimates) ---\n")

fe         <- lme4::fixef(design$lmer_fit)
coef_df    <- coef(design$lmer_fit)[[design$group_name]]
coef_means <- colMeans(coef_df)
coef_vars  <- apply(coef_df, 2L, var)
coef_sds   <- sqrt(coef_vars)
w          <- max(nchar(design$re_coef_names))

for (nm in design$re_coef_names) {
  Xj    <- design$X_hyper[[nm]]
  other <- setdiff(colnames(Xj), "(Intercept)")
  hyper_rhs <- if (length(other) == 0L) "1" else paste(c("1", other), collapse = " + ")

  gamma <- setNames(
    vapply(colnames(Xj), function(col) {
      if (nm == "(Intercept)") {
        if (col %in% names(fe)) unname(fe[col]) else 0
      } else if (col == "(Intercept)") {
        if (nm %in% names(fe)) unname(fe[nm]) else unname(coef_means[nm])
      } else {
        cand <- c(paste0(col, ":", nm), paste0(nm, ":", col))
        hit  <- cand[cand %in% names(fe)]
        if (length(hit)) unname(fe[hit[1L]]) else 0
      }
    }, numeric(1L)),
    colnames(Xj)
  )

  cat(sprintf("  %-*s ~ %s\n", w, nm, hyper_rhs))
  print(gamma)
  cat("\n")
}

## ------------------------------------------------------------------------
## 5. Empirical SD/variance of per-school coefficients vs lmer VarCorr
## ------------------------------------------------------------------------
cat("--- Between-school SD of random coefficients vs lmer VarCorr ---\n")
vc <- as.data.frame(lme4::VarCorr(design$lmer_fit))
cat(sprintf(" %-16s empirical_sd=%7.4f empirical_var=%8.4f lmer_sd=%7.4f lmer_var=%8.4f\n",
            "(Intercept)",
            coef_sds["(Intercept)"], coef_vars["(Intercept)"],
            vc$sdcor[vc$var1 == "(Intercept)" & is.na(vc$var2)][1L],
            vc$vcov[vc$var1  == "(Intercept)" & is.na(vc$var2)][1L]))
for (nm in setdiff(design$re_coef_names, "(Intercept)")) {
```

```
  if (!nm %in% colnames(coef_df)) next
  lmer_row <- vc[vc$var1 == nm & is.na(vc$var2), ]
 cat(sprintf("  %-16s  empirical_sd=%7.4f  empirical_var=%8.4f  lmer_sd=%7.4f  lmer_var=%8.4f\n",
             nm,
             coef_sds[nm], coef_vars[nm],
             if (nrow(lmer_row)) lmer_row$sdcor[1L] else NA_real_,
             if (nrow(lmer_row)) lmer_row$vcov[1L]  else NA_real_))
}

## ---------------------------------------------------------------------------
## 6. lmer refitted on full-rank schools only (same subset as Prior_Setup)
## ---------------------------------------------------------------------------
cat("\n--- lmer refit: full-rank schools only ---\n")
full_rank_schools <- names(design$re_rank)[design$re_rank]
cat(sprintf("  Using %d of %d schools (dropping rank-deficient: %s)\n\n",
            length(full_rank_schools),
            nlevels(design$groups),
            paste(names(design$re_rank)[!design$re_rank], collapse = ", ")))

dat_fr <- subset(dat, school_id %in% full_rank_schools)
dat_fr$school_id <- droplevels(dat_fr$school_id)
fit_fr <- lme4::lmer(form_lmer, data = dat_fr, control = ctrl_bobyqa)
print(summary(fit_fr))

cat("\n--- VarCorr comparison: all schools vs full-rank schools only ---\n")
vc_fr <- as.data.frame(lme4::VarCorr(fit_fr))
cat(sprintf("  %-16s  all_schools_sd=%7.4f  full_rank_sd=%7.4f\n",
            "(Intercept)",
            vc$sdcor[vc$var1 == "(Intercept)" & is.na(vc$var2)][1L],
            vc_fr$sdcor[vc_fr$var1 == "(Intercept)" & is.na(vc_fr$var2)][1L]))
for (nm in setdiff(design$re_coef_names, "(Intercept)")) {
  row_all <- vc[vc$var1 == nm & is.na(vc$var2), ]
  row_fr  <- vc_fr[vc_fr$var1 == nm & is.na(vc_fr$var2), ]
  cat(sprintf("  %-16s  all_schools_sd=%7.4f  full_rank_sd=%7.4f\n",
              nm,
              if (nrow(row_all)) row_all$sdcor[1L] else NA_real_,
              if (nrow(row_fr))  row_fr$sdcor[1L]  else NA_real_))
}

## ===========================================================================
## 7. Optional stress test (NOT the lmerb example model): esl_observed RE
## ===========================================================================
cat("\n\n=== Section 7 (optional): esl_observed added as random slope ===\n\n")

dat_esl <- subset(
  dat,
  complete.cases(dat[, c("esl_observed")])
)
form_esl <- score_ppvt ~
  private_school + title1 + free_reduced_lunch +
  distracted_a1 + distracted_ppvt +
  free_reduced_lunch:distracted_a1 + esl_observed +
  (1 + distracted_ppvt + distracted_a1 + esl_observed || school_id)
```

```
design_esl <- model_setup(form_esl, data = dat_esl, control = ctrl_bobyqa)
print(design_esl)
```

---

pfamily *Prior Family Objects for Bayesian Models*

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 Beta(shape1, shape2) prior on the binomial probability θ and a Binomial(n_i, θ) 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

mu	a prior mean vector for the the modeling coefficients used in several pfamilies
Sigma	a prior variance-covariance matrix for dNormal() and dIndependent_Normal_Gamma().
dispersion	the dispersion to be assumed when it is not given a prior. Should be provided when the Normal prior is for the gaussian(), Gamma(), quasibinomial, or quasipoisson families. The binomial() and poisson() families do not have dispersion coefficients. Omitted or NULL uses the internal default 1 and sets ddef in prior_list (see Details).
Sigma_0	prior variance-covariance on the precision-weighted coefficient scale for dNormal_Gamma() only (Gaussian). Stored in prior_list$Sigma for compatibility with downstream samplers.
shape	The prior shape parameter for the gamma piece (inverse dispersion / precision). When taking defaults from Prior_Setup, use ps$shape with dNormal_Gamma() and dGamma(), and ps$shape_ING with dIndependent_Normal_Gamma() on the Gaussian calibrated path (see Details).
rate	The prior rate parameter paired with shape. With Gaussian Prior_Setup, dNormal_Gamma() and dIndependent_Normal_Gamma() use ps$rate; for dGamma() with fixed beta, prefer ps$rate_gamma when that field is non-NULL (see Details).
max_disp_perc	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.
disp_lower	lower bound truncation for dispersion
disp_upper	upper bound truncation for dispersion
beta	Initial coefficient matrix (1×1); typically set to the prior mean shape1/(shape1+shape2).
Inv_Dispersion	Logical (default TRUE). Controls which of the two Gamma prior roles dGamma() 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 (simfun = rGamma_reg). • FALSE: conjugate prior on the Gamma or Poisson rate β directly (intercept-only, identity link). The posterior is a closed-form Gamma draw (simfun = rGamma_Conjugate_reg).
lik_shape	Known shape parameter $k > 0$ of the Gamma likelihood. Only used when Inv_Dispersion = FALSE and family = Gamma(link = "identity"). 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 Inv_Dispersion = TRUE.

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 **and** simfun**.** 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.

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 conjugacy) 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, and simfun (see Value).

mu / Sigma: the surrogate Normal mean is shape1/(shape1+shape2) and the surrogate variance is the Beta variance shape1*shape2/((shape1+shape2)^2*(shape1+shape2+1)).

Value

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

pfamily — Character string: the constructor name ("dNormal", "dGamma", "dNormal_Gamma", "dIndependent_Normal_Gamma", or "dBeta").

prior_list — Named list of prior hyperparameters. It is passed into simfun when sampling so the relevant low-level routine receives the prior in a fixed list form. Contents depend on the constructor:

dNormal: mu, Sigma, dispersion, and logical ddef (TRUE if dispersion was omitted or NULL, so the default 1 was used; FALSE if set explicitly).

dGamma: shape, rate, beta, Inv_Dispersion, max_disp_perc, disp_lower, disp_upper. When Inv_Dispersion = FALSE, also includes surrogate mu and Sigma (computed from the Gamma prior moments) and lik_shape.

dNormal_Gamma: mu, Sigma (the Sigma_0 precision-weighted input), shape, rate.

dIndependent_Normal_Gamma: mu, Sigma (coefficient-scale covariance), shape, rate, max_disp_perc, disp_lower, disp_upper.

dBeta: shape1, shape2, beta, and surrogate mu and Sigma computed from the Beta prior moments (mu = shape1/(shape1+shape2), Sigma = shape1*shape2/((shape1+shape2)

okfamilies — Character vector of implemented family names for which this pfamily may be used.

plinks — 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.

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 sampling functions used by pfamily constructors.

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.lmebayes_prior_setup` *Build pfamily objects from a Prior_Setup_lmebayes object*

Description

Converts the per-component Block-2 hyperprior parameters stored in a `Prior_Setup_lmebayes` object into a named list of `pfamily` objects, one per random-effect coefficient (e.g. `"(Intercept)"`, slope names).

Usage

```
## S3 method for class 'lmebayes_prior_setup'
pfamily_list(object, ptypes = "dNormal", ...)
```

Arguments

`object`	An object of class `"lmebayes_prior_setup"` as returned by `Prior_Setup_lmebayes`.
`ptypes`	Character: either a single string applied to every random-effect component, or a character vector / list with one string per component. Allowed values are `"dNormal"` and `"dIndependent_Normal_Gamma"`. A vector may be named with the random-effect coefficient names (any order); unnamed vectors are matched positionally against `names(object$prior_list)`.
`...`	Currently ignored.

Details

For each random-effect coefficient k, the prior parameters come from `object$prior_list[[k]]`:

- `"dNormal"`: `dNormal(mu = mu_fixef, Sigma = Sigma_fixef, dispersion = dispersion_fixef)`. The Block-2 dispersion (the random-effect variance τ_k^2) is treated as known.
- `"dIndependent_Normal_Gamma"`: the same `mu` and `Sigma`, plus a Gamma prior on the Block-2 precision $1/\tau_k^2$ calibrated with the same convention as `Prior_Setup`. The per-component effective prior sample size n_0 is taken from `object$n_prior_dispersion[[k]]` (set by `Prior_Setup_lmebayes` via `pwt_dispersion` / `n_prior_dispersion`, derived from `pwt` by default). Then

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

where p_k is the number of Block-2 coefficients for component k (the `shape_ING` convention with the glmbayesCore default rate b_0). Because $rate = \tau_k^2(shape - 1)$, the implied inverse-Gamma prior on the dispersion has mean exactly τ_k^2 for every n_0 and p_k, while small `pwt_dispersion` keeps it deliberately diffuse.

The dispersion prior must not outweigh the data: \eqn{n_0 \le J} (equivalently \code{pwt_dispersion} \eqn{\le 0.5}) is required, mirroring the sampler-side guard in

\code{\link[glmbayesCore]{two_block_rNormal_reg_v2}} (the ING dispersion envelope caps its log-tilt at the data contribution \eqn{J/2}; a prior-dominated calibration would invalidate it).

\code{disp_lower} and \code{disp_upper} default to the 0.01 and 0.99 quantiles of the \emph{limiting posterior} for \eqn{\tau^2_k} -- the weak-prior (\eqn{n_0 \to 0}) limit of the Block-2 posterior Gamma for the precision (glmbayesCore Chapter A12, Theorem 2), \eqn{\Gamma(a_\infty, b_\infty)} with
\deqn{a_\infty = (J+1)/2, \qquad b_\infty = \tau^2_k\,(J-1)/2,}
inverted to a \eqn{\tau^2} interval:
\deqn{disp_lower = 1 / q_{\Gamma}(0.99;\; a_\infty, b_\infty),
\qquad
disp_upper = 1 / q_{\Gamma}(0.01;\; a_\infty, b_\infty).}
Quantiles of the limiting posterior -- rather than of the prior -- make the window independent of \eqn{n_0}: prior quantiles stretch without bound as the dispersion prior weakens (collapsing the envelope acceptance rate), whereas this window covers at least ~98\% of the exact posterior for every \eqn{n_0} (the finite-\eqn{n_0} posterior is strictly more concentrated than the limit) and keeps sampling cost stable. See \code{inst/ING_TRUNCATION_WINDOW.md} for the derivation. The values are computed once by \code{\link{Prior_Setup_lmebayes}} (stored in its \code{ing_prior} field and shown by its print method); this function reads them from the object. During sampling each \eqn{\tau^2_k} draw is hard-truncated to this window (renormalized inverse-CDF), and because both bounds are supplied the window is \emph{fixed} across all inner Gibbs sweeps, rather than re-derived per sweep from a surrogate posterior. \code{\link{lmerb}} and \code{\link{glmerb}} use \code{disp_lower} as the conservative \eqn{\tau^2_k} plug-in for their eigenvalue / TV convergence calibration, so the resulting bound holds over the chain's entire (truncated) dispersion support.

Value

A named list of "pfamily" objects, with names equal to names(object$prior_list) (the random-effect coefficient names).

See Also

Prior_Setup_lmebayes, dNormal, dIndependent_Normal_Gamma

Examples

```
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))
```

```
  ps <- Prior_Setup_lmebayes(
    score_ppvt ~ private_school + (1 | school_id),
    data = dat
  )

  ## All components as dNormal (known Block-2 dispersion)
  pf1 <- pfamily_list(ps)
  print(pf1[["(Intercept)"]])

  ## All components with a Gamma prior on the Block-2 precision
  pf2 <- pfamily_list(ps, ptypes = "dIndependent_Normal_Gamma")
}
```

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 environment(formula), typically the environment from which glm 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 na.action attribute of data, second a na.action setting of options, and third na.fail if that is unset. The factory-fresh default is na.omit. Another possible value is NULL.
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 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 contrasts.arg of model.matrix.default.
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 n_prior is provided and pwt is still a **scalar** and sd was **not** supplied, pwt is set to n_prior / (n_prior + n_effective). If length(pwt) > 1 (including from sd) or sd was supplied, n_prior does **not** overwrite pwt; it is used only as a scalar for Gamma / S_marg steps. If sd is provided, pwt is computed from the prior standard deviations. If none of these are supplied, pwt defaults to pwt_default_low for models with fewer than 14 coefficients, and pwt_default_high otherwise.
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.
n_prior	Optional scalar effective prior sample size (on the n_effective scale). If provided with scalar pwt and without sd, pwt is recomputed from n_prior. With vector pwt or with sd, pwt is left unchanged and n_prior is used for the Gamma prior on precision and related Gaussian calibration only. If missing and pwt is scalar, n_prior = (pwt/(1-pwt))*n_effective.
sd	Optional vector argument with the prior standard deviations for the coefficients

`dispersion` Optional scalar dispersion override (default `NULL`). 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 (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).

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

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

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 $n_{\mathrm{prior}}/(n_{\mathrm{prior}} + n_{\mathrm{effective}}) = \mathrm{pwt}$ (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` $= n_{\mathrm{prior}} \bar{y}_w$ and `rate` $= n_{\mathrm{prior}}$, 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 (`Sigma_{11}` = $\bar{y}_w / n_{\mathrm{prior}}$).

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

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

which under scalar `pwt` and the default calibration reduces to

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

Gaussian Normal–Gamma calibration and S_{marg}

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

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

The marginal quadratic term is

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

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

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

which makes the limiting behavior as $\mathrm{pwt} \to 0$ transparent.

The calibrated dispersion is

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

and the Normal–Gamma hyperparameters are

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

The independent Normal–Gamma shape is

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

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
$$E[\sigma^2 \mid y] = \frac{S_{\text{marg}}}{n_{\text{effective}} - p}.$$
- Posterior covariance
$$\text{Cov}(\beta \mid y) = E[\sigma^2 \mid y]\,\left(\Sigma_0^{-1} + X^\top W X\right)^{-1}.$$

Weak-prior limits (Theorems 2 and 3)

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

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

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:

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

Value

A list of class `"PriorSetup"` with components:

`mu`	Prior mean vector (length equal to the number of coefficients).
`Sigma`	Coefficient-scale prior variance–covariance matrix.
`Sigma_0`	For `family = gaussian()` only: dispersion-independent prior covariance on the precision-weighted coefficient scale (the `Sigma_0` passed to `compute_gaussian_prior`). Under scalar `pwt`, $\Sigma_0^{-1} = \frac{\text{pwt}}{1-\text{pwt}} X^\top W X$.

dispersion	Calibrated dispersion (Gaussian models only), equal to $S_{\text{marg}}/(n_{\text{effective}} - p)$ under the default calibration.
shape	Derived prior Gamma shape parameter for the Normal–Gamma prior on precision (Gaussian only), $(n_{\text{prior}} + k)/2$.
shape_ING	For gaussian() only when shape is available: dedicated shape parameter for dIndependent_Normal_Gamma(), shape $+ p/2$.
rate	Derived prior Gamma rate parameter (Gaussian only), using the calibrated S_{marg}.
rate_gamma	For gaussian() only, when Gaussian calibration runs: prior Gamma rate for dGamma() / fixed-β use, based on $\text{RSS}_w(\beta_\star)$ at the Zellner blend.
coefficients	Named numeric vector of returned coefficient values. For gaussian() with scalar or vector pwt, 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.
model	The model frame used to construct the design matrix (if model = TRUE).
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 $= n_{\text{prior}}$, and beta centered at the weighted sample mean $\bar{y}_w$; pass to dGamma with Inv_Dispersion = FALSE. See Details.

References

Nygren K (2025). "Chapter A12: Technical Derivations for Priors Returned by Prior_ Setup." Vignette in the glmbayes R package. R vignette name: 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)
```

`Prior_Setup_lmebayes` *Prior setup for the two-block Gibbs lmebayes sampler*

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 (`re_rank`) is a diagnostic check only and does not subset the data. Random-effect variances are treated as fixed at their mixed-model estimates. The returned object provides all inputs needed for the two-block Gibbs sampler:

Usage

```
Prior_Setup_lmebayes(
  formula,
  data,
  family = gaussian(),
  pwt = 0.01,
  pwt_dispersion = NULL,
  n_prior_dispersion = NULL,
  intercept_source = c("null_model", "full_model"),
  effects_source = c("null_effects", "full_model")
)

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

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 `family`. Default `gaussian()`. Non-Gaussian families use `glmer` for calibration; `dispersion_ranef` is omitted (analogous to `Prior_Setup` for flat GLMs).
`pwt`	Prior weight(s) in $(0, 1)$. 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: $(1 - \mathrm{pwt})/\mathrm{pwt}$ for a scalar, and elementwise $\sqrt{(1-\mathrm{pwt}_i)/\mathrm{pwt}_i}\,\sqrt{(1-\mathrm{pwt}_j)/\mathrm{pwt}_j}$ for vectors.

`pwt_dispersion` Optional *relative* prior weight(s) in $(0, 1)$ for the Block~2 dispersion (precision) prior, decoupled from `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 `pwt_dispersion` and `n_prior_dispersion` may be supplied; when neither is, the value is derived from `pwt` (the mean across a component's predictors), keeping the 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 `ing_prior` below).

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

`intercept_source`
Character string controlling the prior mean for intercept terms. One of `"null_model"` (default) or `"full_model"`. When `"null_model"`, the prior mean for each component intercept is taken from a null (intercept-only) fit that omits level-2 predictors, making the `Pr(Prior_tail)` tests more informative. When `"full_model"`, the full-model MLE intercept is used.

`effects_source` Character string controlling the prior mean for non-intercept fixed-effect terms. One of `"null_effects"` (default) or `"full_model"`. When `"null_effects"`, the prior mean for all non-intercept coefficients is set to zero (a neutral shrinkage prior). When `"full_model"`, the full-model MLE slopes are used.

`x` Object of class `"lmebayes_prior_setup"`.

`digits` Number of decimal places for numeric output. Default 4.

`...` Ignored.

Details

Block 1 (per-group, independent):

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

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

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):

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

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

Why default calibration depends on classical estimates. `Prior_Setup_lmebayes` anchors each Block~2 mean at the classical mixed-model estimate and scales covariances from `vcov(fit_ref)` by $(1 - \text{pwt})/\text{pwt}$. This requires:

1. Every `X_hyper[[k]]` column maps to a `fixef(fit_ref)` term.
2. Each RE variance τ_k^2 from `fit_ref` is strictly positive.

Value

Object of class `"lmebayes_prior_setup"` with fields:

`formula` Model formula.

`family` Family object.

`pwt` Prior weight(s) used: the scalar as supplied, or the canonical named list of per-predictor weight vectors when a list was supplied.

`pwt_dispersion` Named per-component vector of relative dispersion prior weights (always present; consistent with `n_prior_dispersion` via $w_k = n_k/(n_k + J)$).

`n_prior_dispersion` 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).

`design` Full `model_setup` object (all groups).

`fit_ref` Reference `lmer`/`glmer` fit on all groups (the full-formula fit from `model_setup`).

`dispersion_ranef` Scalar σ^2 for Gaussian models only; `NULL` otherwise.

`Sigma_ranef` Diagonal RE covariance matrix (Block~1).

`prior_list` Named Block~2 prior list per RE coefficient.

`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 glmbayesCore default calibration with $n_0 =$ `n_prior_dispersion`; since `rate` $= \hat{\tau}_k^2($`shape` $-1)$, the implied inverse-Gamma prior on τ_k^2 has mean exactly $\hat{\tau}_k^2$), and the default τ_k^2 truncation window `disp_lower` / `disp_upper`: the 0.01 / 0.99 quantiles of the *limiting posterior* $\Gamma((J+1)/2,\ \hat{\tau}_k^2(J-1)/2)$ – the weak-prior ($n_0 \to 0$) limit of the Block~2 posterior Gamma for the precision (glmbayesCore Chapter A12, Theorem 2; inverted to a τ^2 interval). This window is identical for all n_0, covers $\geq$ ~98\ 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.

`x` invisibly.

See Also

`model_setup`, `Prior_Setup`, `build_mu_all`

Prior_SetupBlock *Prior setup for row-block* lmb / glmbBlock

Description

Runs Prior_Setup on each block subset of the data.

Usage

```
Prior_SetupBlock(
  formula,
  block,
  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.
block	Block partition: factor or vector of length nrow(data) (after model.frame), a column name in data, l2_blocks counts, or a list of row index vectors (see normalize_block).
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 environment(formula), typically the environment from which glm 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 `na.action` attribute of data, second a `na.action` setting of options, and third `na.fail` if that is unset. The `factory-fresh` default is `na.omit`. Another possible value is `NULL`.

`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 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 `contrasts.arg` of `model.matrix.default`.

`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 `n_prior` is provided and `pwt` is still a **scalar** and `sd` was **not** supplied, `pwt` is set to `n_prior / (n_prior + n_effective)`. If `length(pwt) > 1` (including from `sd`) or `sd` was supplied, `n_prior` does **not** overwrite `pwt`; it is used only as a scalar for Gamma / S_marg steps. If `sd` is provided, `pwt` is computed from the prior standard deviations. If none of these are supplied, `pwt` defaults to `pwt_default_low` for models with fewer than 14 coefficients, and `pwt_default_high` otherwise.

`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.

`n_prior` Optional scalar effective prior sample size (on the `n_effective` scale). If provided with scalar `pwt` and without `sd`, `pwt` is recomputed from `n_prior`. With vector `pwt` or with `sd`, `pwt` is left unchanged and `n_prior` is used for the Gamma prior on precision and related Gaussian calibration only. If missing and `pwt` is scalar, `n_prior = (pwt/(1-pwt))*n_effective`.

`sd` Optional vector argument with the prior standard deviations for the coefficients

`dispersion` Optional scalar dispersion override (default `NULL`). 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 (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).

`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 compute_gaussian_prior 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.

Value

A named list of class "block_PriorSetup". Each element is a Prior_Setup result for one block.

See Also

lmbBlock, multi_prior_setup, Prior_Setup_lmebayes, normalize_block

---

rglmerb *The Bayesian Generalized Linear Mixed-Effects Model Distribution*

Description

Bayesian generalized linear mixed-effects model sampler

Usage

```
rglmerb(
  n,
  design,
  prior,
  family = poisson(),
  dispersion_ranef = NULL,
  fixef_start = NULL,
  m_convergence = NULL,
  n_pilot = NULL,
  gap_tol = 0.0196,
  m_convergence_pilot = NULL,
  tv_tol = 0.01,
  mode_gap_max = 1,
  collect_block1 = TRUE,
  seed = NULL,
  verbose = TRUE,
  progbar = FALSE
)
```

Arguments

n	Integer. Number of independent chains in the main stage.
design	A model_setup object.
prior	A lmebayes_prior_setup object.
family	A family object. Default poisson().
dispersion_ranef	Observation-level measurement dispersion σ^2: required positive scalar for family = gaussian(), or a dGamma() pfamily with Inv_Dispersion = TRUE; must be NULL (default) for poisson() and binomial().
fixef_start	Optional named list of Block~2 starting vectors. When NULL, the ICM start is computed inside the Core engine (lmerb_posterior_mean for Gaussian, glmerb_posterior_mode otherwise).
m_convergence	Optional integer inner Gibbs sweeps per main draw.
n_pilot	Optional integer pilot chains (non-Gaussian only); NULL (default) derives from gap_tol; 0L skips pilot.
gap_tol	Mode–mean gap tolerance for deriving n_pilot when n_pilot is NULL (default 0.0196). Set NULL to skip pilot unless n_pilot is explicit. Ignored for Gaussian.
m_convergence_pilot	Optional pilot inner sweeps (non-Gaussian only).
tv_tol	Total variation tolerance for convergence calibration.
mode_gap_max	Pilot sweep calibration when m_convergence_pilot is NULL (non-Gaussian only).
collect_block1	Collect Block~1 coefficients from main chains (non-Gaussian only).
seed	Optional RNG seed (Gaussian path only).
verbose	Print stage headers and diagnostics.
progbar	Progress bars when verbose is FALSE.

Details

Matrix-level sampler for **lmebayes** model_setup objects and prior containers. Routes by response family:

- family = gaussian() delegates to rLMMNormal_reg or rLMMindepNormalGamma_reg when dispersion_ranef is a dGamma() pfamily.
- Non-Gaussian families delegate to rGLMM (sweep-outer engine with optional pilot/main staging).

See glmerb for the formula-level API.

Value

Object of class c("rglmerb", "list") with Block~2 fields in the fixef.* namespace, plus ranef.mode, Prior, design, and family. Non-Gaussian fits may include n_pilot, pilot, and pilot_chisq; Gaussian fits set n_pilot = 0L and omit pilot output.

See Also

glmerb, rLMMNormal_reg, rLMMindepNormalGamma_reg, rGLMM

---

rlmerb *The Bayesian Linear Mixed-Effects Model Distribution*

Description

Bayesian linear mixed-effects model sampler (two-block Gibbs engine)

Usage

```
rlmerb(
  n,
  design,
  prior,
  dispersion_ranef,
  fixef_start = NULL,
  m_convergence = NULL,
  tv_tol = 0.01,
  seed = NULL,
  progbar = TRUE,
  verbose = TRUE,
  print_icm_table = TRUE
)
```

Arguments

n	Integer. Number of stored draws (each draw is one full pass through m_convergence inner Gibbs sweeps).
design	A model_setup object as returned by model_setup, supplying y, Z, groups, X_hyper, group_name, and re_coef_names.
prior	A lmebayes_prior_setup object as returned by .lmebayes_priors_from_pfamily_list (RE / Block~2 structure).
dispersion_ranef	Required observation-level dispersion: a positive scalar σ^2 (fixed) or a dGamma() pfamily with Inv_Dispersion = TRUE for a Gamma prior on σ^2. Typically Prior_Setup_lmebayes(...)$dispersion_ranef for a fixed plug-in.
fixef_start	Optional named list of starting hyper-parameter vectors (one per RE component). When NULL (default), the ICM posterior mean is computed inside the Core engine (rLMMNormal_reg or rLMMindepNormalGamma_reg).
m_convergence	Optional integer. Number of inner Gibbs sweeps per stored draw. When NULL (default), derived from tv_tol via Theorem 3 (Nygren 2020) and floored at the derived m_min. A user-supplied value is floored at m_min with a warning if it had to be raised.

tv_tol	Single numeric in (0, 1). Total variation tolerance used for convergence calibration. Default 0.01.
seed	Optional integer RNG seed. Default NULL.
progbar	Logical. Show a text progress bar during sampling. Default TRUE.
verbose	Logical. Print the lmer-vs-ICM table and the convergence calibration line. Default TRUE.
print_icm_table	Logical. When FALSE, skip the reference-vs-ICM table (e.g. when rglmerb prints glmer-labelled output). The convergence calibration line from the Core engine still follows verbose. Default TRUE.

Details

Full sampling engine for Gaussian linear mixed models, parallel to rlmb in **glmbayes** and rglmerb / glmerb in **lmebayes**. Takes structured design and prior objects, computes the ICM posterior mean internally, and delegates replicate-chain sampling to rLMMNormal_reg or rLMMindepNormalGamma_reg when dispersion_ranef is a dGamma() pfamily.

rlmerb is called internally by lmerb after model_setup and prior construction are complete. It can also be called directly in simulation or Gibbs-sampling workflows where formula parsing and model-fit overhead are unnecessary.

Value

An object of class c("rlmerb", "list") with Block~2 fields in the fixef.* namespace (Core LMM engines): fixef, fixef.mode, fixef.init, fixef.means, fixef.dispersion, fixef.dispersion.mean, fixef.iters, fixef.iters.mean, fixef.mu; Block~1 draws in coefficients; ranef.mode; m_convergence; convergence; Prior; design.

See Also

lmerb, rLMMNormal_reg, rLMMindepNormalGamma_reg, glmerb, rglmerb, 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 *Summarize blmb fits*

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 *Summarize Bayesian linear mixed model fits*

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_overview, fixef (per-RE-component tables), ranef_overview, any_non_normal, tau2 (per-component Block~2 dispersion table: prior type, plug-in value, posterior mean / SD / quantiles from fixef.dispersion for sampled ING components, and Cand/draw, the average number of Block~2 candidates per accepted draw from fixef.iters.mean – 1 for dNormal, roughly the reciprocal acceptance rate of the envelope sampler for ING), and optionally ranef_groups.

See Also

lmerb, glmerb, print.lmerb, summary.glmb, summary.mlmb

Index