Package: BDIB (via r-universe)

October 19, 2026

Type Package

Title Bayesian Dynamic Information Borrowing for Longitudinal Data with Informative Dropout

Version 0.1.1

Description Implements 'BDIB' (Bayesian dynamic information borrowing) for longitudinal outcomes with informative dropout. The package provides the mixture prior for longitudinal data borrowing (MLB) and the self-adapting mixture prior for longitudinal data information borrowing (SLB). Both methods use a shared-parameter model for outcomes and dropout.

License GPL-3

Encoding UTF-8

Depends R (>= 4.0.0)

Imports RBesT, jagsUI, stats

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.0.0

Author Saijun Zhao [aut, cre]

Maintainer Saijun Zhao <saijzhao@iu.edu>

Config/pak/sysreqs cmake make jags libicu-dev

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

Date/Publication 2026-10-18 18:12:27 UTC

RemoteUrl https://github.com/cran/BDIB

RemoteRef HEAD

RemoteSha b633d19021a870ef31ab86561cab4b1a8135ce96

Contents

fit_spm_jags 2
gaussian_kde 4
generate_spm 5
Mixprior 7
MLB_SLB_anlaysis 8
SLB_weight 11
unit_information_prior 12

Index **14**

`fit_spm_jags` *Fit SPM with MNAR monotone dropout in JAGS (continuous or binary outcomes)*

Description

Unified JAGS interface for the shared-parameter model (SPM) with MNAR dropout:

- `family="gaussian"`: Continuous outcome.
- `family="bernoulli"`: Binary outcome.

Usage

```
fit_spm_jags(
  df,
  family = c("gaussian", "bernoulli"),
  prior_theta = NULL,
  params_to_save = NULL,
  n.chains = 3,
  n.iter = 10000,
  n.burnin = 5000,
  n.thin = 2,
  parallel = FALSE
)
```

Arguments

`df`	Long-format `data.frame` with columns `id`, `time`, `R`, `Y`. `R=1` means observed, `R=0` means missing.
`family`	Either `"gaussian"` or `"bernoulli"` (probit).
`prior_theta`	Optional prior for θ. • `family="gaussian"`: `RBesT::normMix` object (mixture Normal on θ). • `family="bernoulli"`: a `"betaMix"` object (mixture Beta on θ). If `NULL`, defaults to a very weak one-component prior.

`params_to_save` Character vector of parameter names to monitor. If `NULL`, sensible defaults are used for each family.

`n.chains`, `n.iter`, `n.burnin`, `n.thin`
MCMC settings passed to `jagsUI::jags`.

`parallel` Logical; whether to run chains in parallel (via `jagsUI`).

Value

Numeric vector of posterior draws of θ.

Examples

```
# ---- Example 1: Continuous outcome with weak prior ----
set.seed(1000)

# 1) Generate longitudinal data with MNAR monotone dropout
t_vec <- seq(4, 48, 4)
gen <- generate_spm(
  n = 80, t_vec = t_vec, family = "gaussian",
  alpha0 = -4, alpha1 = 0.05, rho = 1, sigma_b = 1,
  theta = -3.2,   # total change at the last visit (= slope * max(t))
  sigma_e = 1
)
df <- gen$dat  # long-format data: id, time, R, Y

# 2) Define a weakly informative prior for theta (one Normal component with large variance)
prior <- RBesT::mixnorm(c(1, 0, 1000))  # weight=1, mean=0, sd=1000

# 3) Fit the SPM (MNAR) and obtain posterior draws of theta

# Full MCMC fitting takes longer than a brief example check.
draws_theta <- fit_spm_jags(
  df = df, family = "gaussian",
  prior_theta = prior,
  params_to_save = c("theta","slope","alpha0","alpha1","rho","sigma_e","sigma_b"),
  n.chains = 3, n.iter = 10000, n.burnin = 5000, n.thin = 2, parallel = FALSE
)

# Summarize posterior results
summary(draws_theta)    # posterior mean, median, quantiles
mean(draws_theta)       # posterior mean of theta


# ---- Example 2: Binary outcome (probit link) with weak prior ----
set.seed(1000)
# 1) Generate longitudinal binary data with MNAR monotone dropout
t_vec <- seq(4, 48, 4)
gen <- generate_spm(
  n       = 200,
  t_vec   = t_vec,
  family  = "bernoulli",
  alpha0  = -4,       # dropout logit-intercept
```

```
  alpha1  =  0.05,    # dropout logit-slope in time
  rho     = -1,       # association between random effect and dropout
  sigma_b =  1,       # SD of random intercept b_i
  theta   =  0.4,    # target marginal Pr(Y=1) at final visit
  beta0   = -2        # outcome probit-intercept
)
df <- gen$dat  # long-format data: id, time, R, Y

# 2) Define a weakly informative prior for theta:
#    Beta(1,1) (Uniform on (0,1)) as a one-component mixture
prior <- RBesT::mixbeta(c(1, 1, 1))  # weight=1, a=1, b=1

# 3) Fit the SPM (MNAR) with probit link for the binary outcome
#    and obtain posterior draws of theta (final-visit response rate)

# Full MCMC fitting takes longer than a brief example check.
draws_theta <- fit_spm_jags(
  df             = df,
  family         = "bernoulli",
  prior_theta    = prior,
  params_to_save = c("theta","slope","beta0","alpha0","alpha1","rho","sigma_b"),
  n.chains       = 3,
  n.iter         = 5000,
  n.burnin       = 3000,
  n.thin         = 2,
  parallel       = FALSE
)

# 4) Summarize posterior results
summary(draws_theta)   # posterior mean, median, quantiles of theta
mean(draws_theta)      # posterior mean of theta
```

`gaussian_kde` *Gaussian Kernel Density Estimation*

Description

Gaussian Kernel Density Estimation

Usage

```
gaussian_kde(samples, eval_points, bandwidth = NULL)
```

Arguments

`samples`	Numeric vector of samples.
`eval_points`	Numeric vector of locations to evaluate.
`bandwidth`	Optional bandwidth; Silverman's rule if `NULL`.

Value

Numeric vector of density estimates.

Examples

```
set.seed(1000)
xs <- rnorm(1000, mean = -3, sd = 0.5)
pts <- seq(-4.5, -1.5, length.out = 5)
dens <- gaussian_kde(xs, pts)      # KDE values at 'pts'
cbind(pts, round(dens, 4))
```

---

generate_spm — *Generate longitudinal data from a shared-parameter model (SPM) with MNAR monotone dropout (continuous or binary outcomes)*

Description

Generate longitudinal data from a shared-parameter model (SPM) with MNAR monotone dropout (continuous or binary outcomes)

Usage

```
generate_spm(
  n,
  t_vec,
  family = c("gaussian", "bernoulli"),
  alpha0,
  alpha1,
  rho,
  sigma_b,
  theta,
  beta0 = NA_real_,
  sigma_e = NA_real_
)
```

Arguments

`n`	Integer. Sample size (number of subjects).
`t_vec`	Numeric vector of visit times.
`family`	Character. Either `"gaussian"` for a continuous outcome or `"bernoulli"` for a binary outcome.
`alpha0`, `alpha1`	Numerics. Intercept and time slope in the dropout model (logit link).
`rho`	Numeric. Coupling between the shared random effect b_i and the dropout model; $\rho \neq 0$ induces MNAR.
`sigma_b`	Numeric (>0). Standard deviation of the subject-specific random intercept $b_i \sim \mathcal{N}(0, \sigma_b^2)$.

`theta`	Numeric. Target effect: • `family="gaussian"`: change at final visit $_K$: $\theta = \beta_1 t_K$ (so the slope is $\beta_1 = \theta/t_K$). • `family="bernoulli"`: marginal response probability at the final visit in $(0, 1)$ (probit link).
`beta0`	Numeric. For `family="bernoulli"` only: probit intercept in $\Phi(\beta_0+\beta_1 t_k+b_i)$. Ignored for `"gaussian"`.
`sigma_e`	Numeric (>0). For `family="gaussian"` only: residual SD.

Details

Outcome model (shared-parameter).

- *Gaussian:* $Y_{ik} \mid b_i \sim \mathcal{N}(\beta_1 t_k + b_i, \sigma_e^2)$, with $\beta_1 = \theta/t_K$.
- *Bernoulli (probit):* $Y_{ik} \mid b_i \sim \mathrm{Bernoulli}(\Phi(\beta_0 + \beta_1 t_k + b_i))$, with $\beta_1 = \{\sqrt{1+\sigma_b^2}\,\Phi^{-1}(\theta) - \beta_0\}/t_K$.

Dropout model (MNAR, monotone).

$$\mathrm{logit}\ \Pr(R_{ik} = 0 \mid R_{i,k-1} = 1, b_i) = \alpha_0 + \alpha_1 t_k + \rho b_i,$$

and once dropout occurs ($R_{ik} = 0$), all subsequent $R_{i,k'} = 0$ for $k' > k$.

Value

A list with:

`Ymat`	$n \times K$ matrix of outcomes (NA for missing).
`Rmat`	$n \times K$ matrix of response indicators (1=observed, 0=missing).
`dat`	Long-format `data.frame` with columns `id`, `time`, `R`, `Y`.

Examples

```
set.seed(1000)
t_vec <- seq(4, 48, 4)
# Continuous outcome
out_c <- generate_spm(
  n = 50, t_vec = t_vec, family = "gaussian",
  alpha0 = -4, alpha1 = 0.05, rho = -1, sigma_b = 1,
  theta = -3.2, sigma_e = 1
)
# Binary outcome
out_b <- generate_spm(
  n = 200, t_vec = t_vec, family = "bernoulli",
  alpha0 = -4, alpha1 = 0.05, rho = -1, sigma_b = 1,
  theta = 0.4, beta0 = -2
)
```

`Mixprior` *Constructing a mixture prior for longitudinal data with informative dropout*

Description

The `Mixprior` function constructs a mixture prior for longitudinal data with informative dropout by mixing an informative prior (built from the historical arm/posterior draws) and a non-informative (e.g., unit-information) prior. It supports both continuous (`normMix`) and binary (`betaMix`) outcomes and simply delegates to `RBesT::mixcombine`.

Usage

```
Mixprior(informative_prior, noninformative_prior, weight)
```

Arguments

`informative_prior`
: Informative prior for θ derived from historical data, represented as `RBesT::normMix` (continuous) or `"betaMix"` (binary).

`noninformative_prior`
: Non-informative prior (e.g., unit-information for continuous outcome or Uniform(0,1) for binary outcome) of the same class as `informative_prior`:

- Continuous: `normMix`. See also `unit_information_prior()`.
- Binary: `"betaMix"`; commonly `RBesT::mixbeta(c(1,1,1))`.

`weight`
: Mixture weight assigned to the informative component ($0 \leq$ `weight` ≤ 1); may be a scalar w or a vector `c(w, 1-w)`.

Value

A prior of the same class as `informative_prior`: `normMix` for continuous endpoints or `"betaMix"` for binary endpoints.

Examples

```
# --- Continuous endpoint (normMix) ---
set.seed(1000)
hist_draws <- rnorm(1000, mean = -3.2, sd = 0.35)
inf_prior_c  <- RBesT::mixfit(hist_draws, type = "norm", Nc = 3)  # informative
nf_prior_c   <- unit_information_prior(inf_prior_c, n_hist = 200) # unit-info
w_c <- 0.6  # (e.g., from SLB_weight)
prior_c <- Mixprior(informative_prior   = inf_prior_c,
                    noninformative_prior = nf_prior_c,
                    weight = w_c)
prior_c

# --- Binary endpoint (betaMix) ---
```

```
inf_prior_b <- RBesT::mixbeta(c(1, 20, 30))    # Beta(20,30) as one-component mix
nf_prior_b  <- RBesT::mixbeta(c(1, 1, 1))      # Uniform(0,1)
w_b <- 0.5
prior_b <- Mixprior(informative_prior   = inf_prior_b,
                    noninformative_prior = nf_prior_b,
                    weight = w_b)
prior_b
```

`MLB_SLB_anlaysis` *The mixture prior for longitudinal data borrowing (MLB) and self-adapting mixture prior for longitudinal data information borrowing (SLB) Analysis under SPM with MNAR dropout*

Description

Arms and notation (H/C/T):

- `H` — Historical control arm. External/previous-study control, used to build an *informative prior* for the concurrent control.
- `C` — Concurrent control arm. In-trial control; first fitted under a non-informative prior to assess prior–data conflict and compute an adaptive weight; then re-fitted under an MLB/SLB prior.
- `T` — Treatment arm. In-trial treatment; typically fitted under a non-informative prior, then compared against the (re)fitted control.

Usage

```
MLB_SLB_anlaysis(
  historical_df,
  concurrent_df,
  treatment_df,
  family = c("gaussian", "bernoulli"),
  delta,
  n_hist = NULL,
  fixed_weight = 0.5,
  mcmc_settings = list(n.chains = 3, n.iter = 10000, n.burnin = 5000, n.thin = 2,
    parallel = FALSE),
  decision_MLB,
  direction_MLB = c("lower", "higher"),
  decision_SLB,
  direction_SLB = c("lower", "higher")
)
```

Arguments

`historical_df`	Long-format data.frame for H with columns `id,time,R,Y`.
`concurrent_df`	Long-format data.frame for C with columns `id,time,R,Y`.

`treatment_df` Long-format data.frame for T with columns `id,time,R,Y`.

`family` Character, `"gaussian"` or `"bernoulli"` (probit link), passed to SPM.

`delta` Clinically significant difference (CSD) used in SLB weight.

`n_hist` Positive integer: historical sample size for unit-information prior (continuous only).

`fixed_weight` MLB mixing weight ω (default 0.5).

`mcmc_settings` JAGS settings list with fields `n.chains,n.iter,n.burnin,n.thin,parallel`.

`decision_MLB` A REQUIRED decision function for MLB prior with threshold p that takes two numeric vectors of posterior samples (θ_T, θ_C) and returns logical/0-1 indicating treatment superiority.

Direction of test (how probabilities are computed/decisions typically defined):

- Larger-is-better: `function(theta_t, theta_c, p=0.95) mean(theta_t > theta_c) > p`
- Smaller-is-better: `function(theta_t, theta_c, p=0.95) mean(theta_t < theta_c) > p`

`direction_MLB` Character scalar specifying the clinical direction of superiority for MLB prior:

- `"lower"`: smaller is better; report $\Pr(\theta_T < \theta_C)$.
- `"higher"`: larger is better; report $\Pr(\theta_T > \theta_C)$.

`decision_SLB` A REQUIRED decision function for SLB prior with threshold p that takes two numeric vectors of posterior samples (θ_T, θ_C) and returns logical/0-1 indicating treatment superiority.

Direction of test (how probabilities are computed/decisions typically defined):

- Larger-is-better: `function(theta_t, theta_c, p=0.95) mean(theta_t > theta_c) > p`
- Smaller-is-better: `function(theta_t, theta_c, p=0.95) mean(theta_t < theta_c) > p`

`direction_SLB` Character scalar specifying the clinical direction of superiority for SLB prior:

- `"lower"`: smaller is better; report $\Pr(\theta_T < \theta_C)$.
- `"higher"`: larger is better; report $\Pr(\theta_T > \theta_C)$.

Details

Target parameters $\theta_H, \theta_C, \theta_T$:

- `family = "gaussian"` (continuous outcome): θ is the total change at the final visit t_K, i.e., $\theta = \beta_1 t_K$ (thus $\beta_1 = \theta / t_K$).
- `family = "bernoulli"` (binary, probit link): θ is the marginal response probability at the final visit, $\Pr(Y = 1 \text{ at } t_K) \in (0, 1)$.

We write θ_H, θ_C, θ_T for the historical, concurrent-control, and treatment arms, respectively.

Workflow (continuous or binary endpoints):

1. Fit **H** under SPM with a vague prior on θ to obtain posterior draws.
2. Build an *informative prior* g_0 for θ_C via `RBesT::mixfit` on the θ_H draws.
3. Build a *non-informative prior* g_1: Gaussian: unit-information prior from g_0 (inflate SD by $\sqrt{n_{\text{hist}}}$); Bernoulli: $\text{Beta}(1, 1)$.
4. Fit **C** under g_1 to get posterior samples.
5. Compute SLB weight via posterior-probability-ratio KDE (see `SLB_weight`).
6. Construct MLB prior (fixed weight) and SLB prior (data-adaptive weight) via `Mixprior`.
7. Fit **C** under MLB/SLB priors; fit **T** under g_1.
8. Posterior comparisons θ_T vs θ_C and decisions per `direction`.

Value

An object of class `"mlbslb_result"`:

- `estimates`: posterior means of θ for each arm/prior (MLB/SLB).
- `weights`: list with `MLB_weight` and `SLB_weight`.
- `probabilities`: posterior superiority probabilities (per `direction`) under MLB/SLB.
- `decisions`: 0/1 decisions under MLB/SLB according to your `decision()`. (1 indicates test superiority is established).
- `posterior_samples`: posterior draws used in the analysis.

Examples

```
set.seed(1000)
t_vec <- seq(4, 48, 4)
# --- Continuous outcome ---
data_H <- generate_spm(n=60, t_vec, family="gaussian",
                       alpha0=-4, alpha1=0.05, rho=-1, sigma_b=1,
                       theta= -3.2, sigma_e=1)$dat
data_C <- generate_spm(n=30,  t_vec, family="gaussian",
                       alpha0=-4, alpha1=0.05, rho=-1, sigma_b=1,
                       theta= -3.2, sigma_e=1)$dat
data_T <- generate_spm(n=60,  t_vec, family="gaussian",
                       alpha0=-5.3, alpha1=0.05, rho=-1.5, sigma_b=1,
                       theta= -3.7, sigma_e=1)$dat


# Full MCMC fitting takes several minutes.
ans_1 <- MLB_SLB_anlaysis(
  historical_df = data_H, concurrent_df = data_C, treatment_df = data_T,
  family="gaussian", delta=0.5, n_hist=length(unique(data_H$id)), fixed_weight=0.5,
 mcmc_settings = list(n.chains=3, n.iter=10000, n.burnin=5000, n.thin=2, parallel=FALSE),
  decision_MLB = function(theta_t, theta_c, p = 0.95) mean(theta_t < theta_c) > 0.95,
  # Smaller-is-better (treatment < control), p = 0.95
  direction_MLB =  "lower",
  decision_SLB = function(theta_t, theta_c, p = 0.95) mean(theta_t < theta_c) > 0.95,
  # Smaller-is-better (treatment < control), p = 0.95
  direction_SLB =  "lower"
```

```
)
ans_1$decisions; ans_1$probabilities


# --- Binary outcome ---
set.seed(1000)
data_H <- generate_spm(n=200, t_vec, family="bernoulli",
                       alpha0=-4, alpha1=0.05, rho=-1, sigma_b=1,
                       theta=0.40, beta0=-2)$dat
data_C <- generate_spm(n=100, t_vec, family="bernoulli",
                       alpha0=-4, alpha1=0.05, rho=-1, sigma_b=1,
                       theta=0.40, beta0=-2)$dat
data_T <- generate_spm(n=200, t_vec, family="bernoulli",
                       alpha0=-5.3, alpha1=0.05, rho=-1.5, sigma_b=1,
                       theta=0.50, beta0=-1.5)$dat


# Full MCMC fitting takes several minutes.
ans_2 <- MLB_SLB_anlaysis(historical_df = data_H, concurrent_df = data_C, treatment_df = data_T,
   family="bernoulli", delta=0.1, n_hist=length(unique(data_H$id)), fixed_weight=0.5,
  mcmc_settings = list(n.chains=3, n.iter=10000, n.burnin=5000, n.thin=2, parallel=FALSE),
   decision_MLB = function(theta_t, theta_c, p = 0.95) mean(theta_t > theta_c) > 0.95,
   # Larger-is-better (treatment > control), p = 0.95
   direction_MLB =  "higher",
   decision_SLB = function(theta_t, theta_c, p = 0.95) mean(theta_t > theta_c) > 0.95,
   # Larger-is-better (treatment > control), p = 0.95
   direction_SLB =  "higher"
)
ans_2$decisions; ans_2$probabilities
```

`SLB_weight` *Self-adapting mixture weight for longitudinal borrowing (SLB)*

Description

Given posterior samples of the concurrent control parameter (under a non-informative prior) and a historical point estimate, compute the SLB weight via a kernel-density posterior probability ratio:

$$R = \frac{\pi(\theta = \theta_h \mid \mathcal{D}_C)}{\max\{\pi(\theta = \theta_h + \delta \mid \mathcal{D}_C), \pi(\theta = \theta_h - \delta \mid \mathcal{D}_C)\}}, \quad w = \frac{R}{1+R}.$$

Usage

```
SLB_weight(theta_cc_np_samples, theta_h, delta)
```

Arguments

`theta_cc_np_samples`
: numeric vector. Posterior draws of the concurrent control parameter under a non-informative prior.

theta_h	numeric scalar. Historical estimate.
delta	positive numeric. Clinically significant difference (CSD).

Value

numeric scalar in $[0, 1]$.

Examples

```
set.seed(1000)
# Suppose these are posterior theta samples from concurrent control (NP fit)
theta_cc_np_samples <- rnorm(2000, mean = -3.2, sd = 0.25)

# Historical estimate of theta and clinical delta
theta_h   <- -3.2
delta  <- 0.5

SLB_weight(theta_cc_np_samples, theta_h = theta_h, delta = delta)
```

---

unit_information_prior

Unit-Information Prior for continuous outcomes

Description

Constructs a unit-information (weak) prior from an informative `normMix` by inflating each component's standard deviation by $\sqrt{n_h}$.

Usage

```
unit_information_prior(informative_prior, n_hist)
```

Arguments

informative_prior	A `normMix` (from `RBesT`) built from historical data.
n_hist	Positive integer: historical sample size n_h.

Value

A `normMix` with inflated standard deviations.

Examples

```
# Start with an informative prior from historical samples via RBesT::mixfit
set.seed(1000)
fake_hist_draws <- rnorm(300, mean = -3.2, sd = 0.4)
inf_prior <- RBesT::mixfit(fake_hist_draws, type = "norm", Nc = 3)

nf_prior <- unit_information_prior(inf_prior, n_hist = 300)
# Compare SDs before/after
rbind(
  informative_sd  = as.numeric(inf_prior[3, ]),
  unit_info_sd    = as.numeric(nf_prior[3, ])
)
```

Index

fit_spm_jags, 2

gaussian_kde, 4
generate_spm, 5

Mixprior, 7, *10*
MLB_SLB_anlaysis, 8

SLB_weight, *10*, 11

unit_information_prior, 12