Package: BayesPIM (via r-universe)

November 2, 2026

Title Bayesian Prevalence-Incidence Mixture Model

Version 2.0.1

Description Models time-to-event data from interval-censored screening studies. It accounts for latent prevalence at baseline and incorporates misclassification due to imperfect test sensitivity. For usage details, see the package vignette ``BayesPIM_intro''. Further details can be found in Klausch, Lissenberg-Witte and Coupé (2026) <doi:10.1002/sim.70433>.

License MIT + file LICENSE

URL https://github.com/thomasklausch2/BayesPIM

BugReports https://github.com/thomasklausch2/BayesPIM/issues

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

LinkingTo Rcpp

Imports Rcpp, MASS, doParallel, foreach, parallel, actuar, posterior, survival, flexsurv, graphics, grDevices, stats, utils

Depends R (>= 3.5.0), coda

Suggests knitr, rmarkdown, bookdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev

Repository https://thomasklausch2.r-universe.dev

Date/Publication 2026-11-02 19:06:23 UTC

RemoteUrl https://github.com/thomasklausch2/bayespim

RemoteRef HEAD

RemoteSha 301bcc8a789a0ffe7cf86b5219b5ce7ef07dc8a7

Contents

bayespim 2
gen_data 11
get_ic 14
log_aft_prior 15
mod 17
plot.bayespim 18
plot.ppCIF 19
ppCIF 20
search_prop_sd 22
summary.bayespim 24
trim_mcmc 25

Index **26**

`bayespim` *Fitting Bayesian Prevalence-Incidence Mixture Model*

Description

Estimates the prevalence-incidence mixture model of Klausch et al. (2026) using a Bayesian Gibbs sampler. The model is formulated as an interval-censored survival model over successive intervals, with the possibility of missed events due to imperfect test sensitivity. In addition, baseline tests at time zero may fail to detect pre-study events (prevalence).

Usage

```
bayespim(
  v_obs,
  x_t = NULL,
  x_g = NULL,
  r = NULL,
  dist = "weibull",
  kappa = NULL,
  update_kappa = FALSE,
  kappa_prior = NULL,
  ndraws = 1000,
  warmup = floor(ndraws/2),
  warmup_updated = FALSE,
  prop_sd = NULL,
  slice_width = 1,
  chains,
  seed_chains = NULL,
  save_every = 1,
  update_till_converge = FALSE,
  maxit = Inf,
  max_rhat = 1.01,
```

```
  min_effss = chains * 100,
  sampler = "slice_collapsed",
  log_prior_fun = log_aft_prior,
  beta_prior = "norm",
  tau_t = 1,
  sig_prior = 1,
  tau_g = 1,
  fix_sigma = FALSE,
  q_prior_sd = 1,
  fix_q = FALSE,
  prev_run = NULL,
  ndraws_update = NULL,
  prev = TRUE,
  par_exp = FALSE,
  rescale_times = TRUE,
  standardize_covariates = TRUE,
  ini_spread = 0.5,
  silent = FALSE
)
```

Arguments

v_obs	A list of length n of numeric vectors representing screening times. The first element of each vector should always be `0` and the last element `Inf` in the case of right censoring.
x_t	A numeric matrix of dimension $n \times p_t$ containing covariates for the AFT latent-time model. Missing values are not allowed. Categorical variables must be dummy-coded.
x_g	A numeric matrix of dimension $n \times p_g$ containing covariates for the probit prevalence model. Missing values are not allowed. Categorical variables must be dummy-coded.
r	A binary vector of length n indicating whether a baseline test was conducted (1 for yes, `0` for no / missing baseline test).
dist	Character. Distribution for the time-to-incidence variable. Supported choices are `"weibull"`, `"lognormal"`, `"loglog"` (log-logistic), `"gamma"`, and `"gengamma"` (Prentice generalized gamma). The generalized-gamma model is supported only with `sampler = "slice_collapsed"`.
kappa	Numeric or `NULL`. Fixed test sensitivity used when `update_kappa = FALSE`; it must then be specified and lie in $(0, 1]$. When `update_kappa = TRUE`, this argument is ignored because each chain receives internally generated starting values for κ.
update_kappa	Logical. If `TRUE`, the test sensitivity (κ) is updated during the Gibbs sampler.
kappa_prior	A numeric vector `c(mean, sd)` specifying the mean and standard deviation of a Beta prior for κ. The mean must lie strictly between zero and one, and `sd` must be positive and smaller than $\sqrt{\text{mean}(1 - \text{mean})}$. Malformed or infeasible values produce an error. If `NULL`, a warning is issued and the uniform $\text{Beta}(1, 1)$ prior is used.

`ndraws` Integer. The total number of MCMC draws for the main Gibbs sampler.

`warmup` Integer. The number of initial generated MCMC iterations per chain omitted before posterior summaries and convergence diagnostics. This is always interpreted on the generated-iteration scale, independently of `save_every`. Defaults to half of `ndraws`, rounded down, and must be smaller than `ndraws` for an initial run.

`warmup_updated` Logical. If `TRUE`, each update adds the current warmup increment to the warmup stored in `prev_run`. If `FALSE`, the stored warmup is retained.

`prop_sd` Numeric. The standard deviation for the proposal (jumping) distribution used when `sampler = "mh"`. It can be searched for heuristically using search_prop_sd and is not used by either slice sampler.

`slice_width` Numeric. The initial bracket width used by the univariate slice samplers. It affects computational efficiency but not the stationary distribution and is not used by `sampler = "mh"`.

`chains` Integer. The number of MCMC chains to run.

`seed_chains` Optional integer vector with one unique seed per chain. For a new fit, each seed initializes that chain's starting values and Gibbs sampler. If `NULL`, unique seeds are generated randomly. When `prev_run` is supplied, the saved end-of-chain RNG states are restored and `seed_chains` is ignored.

`save_every` Positive integer. Store the parameter state from every `save_every`-th generated iteration. The default, `1`, stores every draw. Values greater than one reduce the memory occupied by the returned chains and by chains accumulated during automatic updating, but permanently discard intervening parameter draws. Convergence diagnostics and summaries use every stored post-warm-up draw and never thin again. This setting is inherited by continued runs.

`update_till_converge` Logical. If `TRUE`, the model is updated iteratively until R-hat is at most `max_rhat` and effective sample size is at least `min_effss` for every sampled parameter. Diagnostics are calculated with the `posterior` package and printed whether this argument is `TRUE` or `FALSE`, unless `silent = TRUE`.

`maxit` A positive whole number or `Inf`. The maximum number of MCMC draws allowed before interrupting automatic convergence updates. `maxit` does not truncate the initially requested `ndraws`; a warning is given if an initial run requests more draws than a finite `maxit`. Default is `Inf` (i.e., no automatic interruption).

`max_rhat` Numeric. The maximum rank-normalized split R-hat accepted for every sampled parameter during convergence assessment and automatic updating. Defaults to `1.01`.

`min_effss` Integer. The minimum effective sample size required for each parameter before convergence is accepted during iterative updating.

`sampler` Character. Incidence-parameter update method. Use `"mh"` for Metropolis-Hastings, `"slice"` for univariate slice sampling conditional on augmented exact incidence times, or `"slice_collapsed"` to sample only the latent screening interval and update the incidence parameters with the interval-censored likelihood. The default is `"slice_collapsed"`. The generalized-gamma model requires this sampler.

`log_prior_fun` Function used to evaluate the log-prior for the AFT incidence parameters. It must accept the named arguments `eta`, `dist`, `beta_prior`, `tau_t`, `sig_prior`, and `q_prior_sd`, and return one numeric log-density value. For `dist = "gengamma"`, `eta` ends in `c(log(sigma), Q)`; otherwise it ends in `log(sigma)`.

`beta_prior` Character. Specifies the type of prior for the latent-time regression coefficients (β_{tj}); options are `'norm'` for normal and `'t'` for student-t.

`tau_t` Numeric. The hyperparameter for the prior distribution of the regression coefficients (β_{tj}) in the AFT latent-time model. For a normal prior, this is the standard deviation; for a student-t prior, it represents the degrees of freedom. The default produces a standard-normal prior.

`sig_prior` Numeric. Positive standard deviation of the half-normal prior on the family scale/dispersion parameter σ. For `dist = "gamma"`, σ is the conditional coefficient of variation. When `fix_sigma = TRUE`, this is instead the fixed value of σ.

`tau_g` Numeric. The hyperparameter (standard deviation) for the normal prior distribution of the regression coefficients (β_{gj}) in the probit prevalence model. The default produces a standard-normal prior.

`fix_sigma` Logical. If `TRUE`, the family scale/dispersion parameter σ is fixed at the value supplied through `sig_prior`; if `FALSE`, it is updated.

`q_prior_sd` Positive numeric standard deviation of the zero-centered normal prior on the signed Prentice generalized-gamma shape parameter Q. When `fix_q = TRUE`, Q is instead fixed at this value. It is otherwise ignored for distributions without Q.

`fix_q` Logical. If `TRUE`, fix the generalized-gamma shape parameter Q at `q_prior_sd`; if `FALSE`, update Q. This argument applies only when `dist = "gengamma"` and `sampler = "slice_collapsed"`.

`prev_run` Optional. An unmodified object of class `bayespim` returned by a previous run. The complete fitted state is required, including the parameter chains, the terminal parameters, the latent prevalence and interval states, the covariate scaling, and the saved RNG state; a missing or malformed component is reported as an error before sampling begins. Data, sampler, priors, and RNG state are adopted from the previous run, and each chain continues from its last draw.

`ndraws_update` Integer greater than or equal to 2. The number of MCMC draws for updating a previous run or for convergence updates. If unspecified, `ndraws` is used.

`prev` Logical. If `TRUE`, prevalence adjustment is applied; if `FALSE`, prevalence is assumed to be zero.

`par_exp` Logical. If `TRUE`, the parameter expansion technique of Liu & Wu (1999) with a Haar prior is employed for updating the regression coefficients (β_{wj}) in the prevalence model. Experimental: tests suggest that it does not improve convergence or reduce autocorrelation.

`rescale_times` Logical. If `TRUE`, screening times are rescaled internally by the median latest finite observation time to improve numerical stability; returned times and incidence parameters are restored to the original scale.

`standardize_covariates`
: Logical. If `TRUE`, the default, non-binary columns of `x_t` and `x_g` are centered at their sample means and divided by their sample standard deviations for fitting. Columns with exactly two observed values are treated as binary/dummy variables and left unchanged. Returned coefficients are transformed back to the original covariate scale. The coefficient priors therefore apply on the standardized scale. This setting and the fitted centers and scales are inherited by continued runs.

`ini_spread`
: Numeric. Lower endpoint of the deterministic scaling factors used to disperse fresh-chain starting coefficients. With multiple chains, factors are equally spaced from `ini_spread` to `1`, where `0` initializes coefficients at zero and `1` uses the fitted initialization estimates. With one chain, the fitted estimates are used directly. Values must lie between `0` and `1`.

`silent`
: Logical. If `TRUE`, suppress the progress information printed while the model is fitted and updated, namely the notice that a previous run is being updated and the convergence diagnostics reported after each set of draws. Warnings and errors are still signaled, and the diagnostics remain available in the `convergence` element of the returned object. Unlike most arguments, `silent` is never inherited from `prev_run`: it applies only to the current call, so an update of a silently fitted model prints again unless `silent = TRUE` is supplied afresh.

Details

This Bayesian prevalence-incidence mixture model (PIM) characterizes time to incidence through accelerated failure time (AFT) scaling. With $\eta_i = \boldsymbol{x}'_{ti}\boldsymbol{\beta}_t$, covariates multiply event times by $\exp(\eta_i)$ relative to the corresponding baseline distribution. For the Weibull, lognormal, and log-logistic families this is the familiar log-location-scale representation

$$\log(t_i) = \eta_i + \sigma_t \epsilon_i.$$

For `dist = "gamma"`, BayesPIM instead uses

$$t_i \mid \boldsymbol{x}_{ti} \sim \text{Gamma}\{\sigma_t^{-2}, \sigma_t^{-2}\exp(-\eta_i)\},$$

so $E(t_i \mid \boldsymbol{x}_{ti}) = \exp(\eta_i)$ and σ_t is the conditional coefficient of variation. For `dist = "gengamma"`, BayesPIM uses the Prentice generalized gamma with location η_i, scale σ_t, and signed shape Q. Generalized-gamma fitting is available only with `sampler = "slice_collapsed"`. The covariate matrix for the latent-time model is supplied through `x_t`.

Baseline prevalence is modeled using a probit formulation $Pr(g_i = 1|\boldsymbol{x}_{gi}) = Pr(w_i > 0|\boldsymbol{x}_{gi})$ with

$$w_i = \boldsymbol{x}'_{gi}\boldsymbol{\beta}_g + \psi_i$$

where ψ_i follows a standard normal distribution, and the covariate vector $\boldsymbol{x}_{gi}$ is given in the `x_g` matrix. The latent continuous probit variable w_i determines the modeled binary prevalence variable: $g_i = 1$ if $w_i > 0$ and $g_i = 0$ otherwise.

The argument `v_obs` provides the observed testing times for all individuals. It is a list of numeric vectors, where each vector starts with `0` (representing the baseline time) and is followed by one or

more screening times. The final entry is `Inf` in the case of right censoring or indicates the time of a positive test if an event is observed. Specifically:

- If the baseline test is positive, the vector consists solely of `c(0)`.
- If the baseline test is negative and right censoring occurs before the first regular screening, the vector is `c(0, Inf)`.
- Otherwise, the vector ends with `Inf` in the case of right censoring (e.g., `c(0, 1, 3, 6, Inf)`) or ends at the event time (e.g., `c(0, 1, 3, 6)` for an event detected at time `6`).

By convention, every vector in `v_obs` starts with `0`. However, the binary vector `r` of `length` n indicates whether the baseline test was conducted (`r[i] = 1`) or missing (`r[i] = 0`) for each individual `i` in `v_obs`. For further details on coding, see Section 2 of the main paper.

Test sensitivity can be fixed to a value `kappa` by setting `update_kappa = FALSE`, or it can be estimated if `update_kappa = TRUE`. When estimated with `kappa_prior = c(m, s)`, a Beta prior with mean m and standard deviation s is used. Its shape parameters are calculated analytically as

$$\nu = \frac{m(1-m)}{s^2} - 1, \qquad a = m\nu, \qquad b = (1-m)\nu.$$

The requested mean must lie strictly between zero and one, and the standard deviation must satisfy $0 < s < \sqrt{m(1-m)}$. Malformed or infeasible specifications stop with an informative error. The calculated shape parameters must be finite and positive. If `kappa_prior = NULL`, the function warns and uses an uninformative $\mathrm{Beta}(1, 1)$ prior. In general, we advise against using an uninformative prior, but this default avoids favoring any specific informative prior.

The Gibbs sampler runs for `ndraws` iterations for each of `chains` total chains. Incidence parameters can be updated by Metropolis-Hastings, by univariate slice sampling conditional on augmented exact incidence times, or by collapsed univariate slice sampling conditional only on sampled screening intervals. The collapsed slice sampler (`sampler = "slice_collapsed"`) is the default because it has shown lower autocorrelation and faster convergence than the two exact-time samplers. This differs from the original implementation described by Klausch et al., which used a Metropolis-Hastings sampler. The collapsed sampler also supports the generalized-gamma model, for which convergence with the other samplers is typically slow; consequently, `dist = "gengamma"` is available only with the collapsed sampler.

When `sampler = "mh"`, the Metropolis step applies a normal proposal distribution with standard deviation `prop_sd`, which must be selected by trial and error. An optimal acceptance rate is approximately 23%, which can be computed per MCMC run from the model output. The function search_prop_sd provides a heuristic for selecting an effective proposal standard deviation.

We recommend running at least two chains to facilitate standard MCMC diagnostics such as the Gelman-Rubin statistic. For larger analyses, users may run more chains, but should ensure that their computing environment permits the requested parallel workers. CRAN examples use small sequential runs. Additionally, we suggest first running the sampler for a moderate number of iterations to assess its behavior before using the updating functionality in `prev_run` to extend sampling (see below).

The option `update_till_converge = TRUE` allows `bayespim` to run until convergence. Convergence is achieved when the rank-normalized split R-hat calculated by `posterior::rhat()` is at most `max_rhat` for every sampled parameter and the minimum effective sample size calculated by `posterior::ess_mean()` reaches `min_effss`. The diagnostics use every stored post-warm-up

draw; no additional diagnostic thinning is applied. The current diagnostics are always calculated, stored, and printed. If automated updating is enabled, the sampler continues updating and printing diagnostics until convergence is attained or `maxit` is reached.

Setting `silent = TRUE` suppresses this printing for both forms of updating, the automated loop just described and a manual update requested through `prev_run`. Only the progress output is withheld: the diagnostics are still calculated and stored in the `convergence` element of the returned object, and warnings and errors are still signaled.

The priors for the regression coefficients and distributional parameters can be controlled using `beta_prior`, `tau_t`, `sig_prior`, `tau_g`, and `q_prior_sd`. Specifically:

- `beta_prior` determines the prior type for β_{tj} (either `normal` or Student-t `t`).
- `tau_t` specifies either the standard deviation (for normal priors) or degrees of freedom (for Student-t priors). The default is a standard normal prior.
- A half-normal prior is used for the positive family scale/dispersion parameter σ, with `sig_prior` controlling the standard deviation. In the gamma model, σ is the conditional coefficient of variation.
- A zero-centered normal prior is assigned to β_{gj}, with `tau_g` controlling its standard deviation (default: standard normal).
- In the generalized-gamma model, a zero-centered normal prior is assigned to the signed shape parameter Q, with standard deviation `q_prior_sd`.

Sometimes model fitting can be improved by fixing the σ parameter to a value, which is achieved through setting `fix_sigma = TRUE`. Then, the value specified as `sig_prior` is regarded as the fixed value of σ. For the gamma model this fixes the conditional coefficient of variation. The functionality can also be used to obtain the exponential distribution, akin to a Markov model. For this choose `dist = "weibull"`, `sig_prior = 1`, and `fix_sigma = TRUE`.

The `prev_run` argument allows updating a previous run with additional MCMC draws. The MCMC chain resumes from the last draws, continues, and merges with the original run. If an initial model was fit using `mod <- bayespim(...)`, it can be updated using `mod_update <- bayespim(prev_run = mod)`. By default, `ndraws` additional iterations are added unless otherwise specified via `ndraws_update`. If `warmup_updated = TRUE`, the warmup increases with each update; otherwise the initial warmup is retained.

The Gibbs sampler requires starting values. Incidence-model coefficients are anchored at estimates from an interval-censored survival model, while prevalence-model coefficients are anchored at a penalized probit fit. With multiple chains, the two coefficient vectors are multiplied by equally spaced factors from `ini_spread` to 1; a single chain uses the unscaled fitted values. Each complete candidate state, including the latent prevalence indicators and incidence intervals or times, must have a finite posterior. Each chain makes up to ten initialization attempts. The first eight redraw the latent states at that chain's own spread factor; the ninth uses the fitted estimates unscaled; and the tenth uses zero-centered coefficients. If none of the ten yields a finite posterior, initialization stops with an informative error.

Value

A list containing the following elements:

`ini` A matrix with one row per chain holding the starting values of the original fit. Coefficients are on the returned covariate and time scale, but the latent-time scale parameter is stored as $\log(\sigma_t)$ rather than σ_t. A continued run retains the starting values of the run it continues.

`par` An untrimmed `mcmc.list` containing the parameter draws selected by `save_every`. With the default `save_every = 1`, all generated draws are stored.

`terminal_par` A matrix containing the exact terminal parameter state for each chain. It is used to continue a run when the terminal iteration is not one of the iterations selected by `save_every`.

`terminal_par_internal`
The corresponding terminal state on the internal standardized-covariate and rescaled-time parameterization. It is retained so updates can continue without reconstructing the sampler state.

`times` The terminal augmented latent times t_i for each chain. This is `NULL` for `sampler = "slice_collapsed"`, which does not augment exact latent times. For exact-time samplers, entries with terminal `g == 1` are inactive stored values and should not be interpreted as current latent incidence-time draws.

`k` The terminal latent interval index for every subject and chain. Entries with terminal `g == 0` are the active sampled incidence intervals.

`g` The terminal binary prevalence variables g_i for every subject, concatenated by chain.

`ac` For `sampler = "mh"`, a matrix with stored MCMC draws in rows and chains in columns, where each entry indicates acceptance (`1`) or rejection (`0`). This is `NULL` for either slice sampler.

`ac_cur` For a continued run, the acceptance matrix from the most recent update only. This is `NULL` for either slice sampler and is absent from a fresh fit.

`dat` A data frame containing the last observed interval.

`priors` A list of prior specifications for the model parameters, including `tau_t`, `sig_prior`, `tau_g`, and `q_prior_sd`.

`warmup` The final warmup cutoff used by convergence diagnostics and, by default, the summary and plot methods.

`warmup_updated` Whether warmup is increased during model updates.

`seed_chains` The unique seeds used to initialize the chains in the original fit.

`rng_state` A list containing the RNG state saved at the end of each chain. These states are restored when the fit is updated.

`save_every` The interval at which parameter draws were stored.

`total_iterations`
The total number of iterations generated per chain, including iterations not stored when `save_every > 1`.

`covariate_scaling`
The centers, standard deviations, binary-column indicators, and standardization indicators used for `x_t` and `x_g`.

`runtime` The total runtime of the MCMC sampler.

`max_rhat` The maximum R-hat threshold used for convergence assessment and automatic updating.

`convergence` A list containing the latest parameter-wise R-hat and ESS values, a printable diagnostics table, the convergence criteria, the number of draws and chains assessed, excluded fixed parameters, and the convergence status.

The returned list has class `"bayespim"`. Additionally, most input arguments are returned as part of the output for reference.

References

T. Klausch, B. I. Lissenberg-Witte, and V. M. H. Coupé (2026). "A Bayesian prevalence-incidence mixture model for screening outcomes with misclassification.", Statistics in Medicine, 45(8-9), e70433. doi:10.1002/sim.70433

J. S. Liu and Y. N. Wu, "Parameter Expansion for Data Augmentation," Journal of the American Statistical Association, vol. 94, no. 448, pp. 1264–1274, 1999, doi:10.2307/2669940.

See Also

`summary.bayespim()` and `plot.bayespim()`

Examples

```
# A deliberately short fit for illustrating the interface.
# Use substantially more draws for scientific inference.
set.seed(2025)
dat <- gen_data(kappa = 0.7, n = 100, theta = 0.2,
                p = 1, p_discrete = 1,
                beta_t = c(0.2, 0.2), beta_g = c(0.2, 0.2),
                v_min = 20, v_max = 30, mean_rc = 80,
                sigma_t = 0.2, mu_t = 5, dist = "weibull",
                prob_r = 1)

fit <- bayespim(
  v_obs = dat$v_obs,
  x_t = dat$x,
  x_g = dat$x,
  r = dat$r,
  kappa = 0.7,
  update_kappa = FALSE,
  ndraws = 100,
  warmup = 10,
  chains = 2,
  seed_chains = c(202501, 202502),
  update_till_converge = FALSE,
  sampler = "slice_collapsed",
  dist = "weibull",
  silent = TRUE
)
fit$runtime
```

gen_data *gen_data: Simulate Screening Data for a Prevalence-Incidence Mixture Model*

Description

Generates synthetic data according to the Bayesian prevalence-incidence mixture (PIM) framework of Klausch et al. (2026) with interval-censored screening outcomes. The function simulates continuous or discrete baseline covariates, event times from one of several parametric families, and irregular screening schedules, yielding interval-censored observations suitable for testing or demonstrating PIM-based or other interval-censored survival methods.

Usage

```
gen_data(
  kappa = 0.7,
  n = 1000,
  p = 2,
  p_discrete = 0,
  rho = 0,
  s = 1,
  sigma_t = 1/2,
  mu_t = 4,
  beta_t = NULL,
  beta_g = NULL,
  theta = 0.15,
  v_min = 1,
  v_max = 6,
  mean_rc = 40,
  dist = "weibull",
  q = 1,
  sel_mod = "probit",
  prob_r = 0
)
```

Arguments

kappa	Numeric. Test sensitivity parameter κ used when generating misclassification. A value of 1 implies perfect sensitivity.
n	Integer. Sample size.
p	Integer. Number of continuous multivariate normal baseline covariates to simulate.
p_discrete	Integer. If 1, include an additional discrete covariate X_{discrete} from $\text{Bernoulli}(0.5)$; otherwise, none.
rho	Numeric. A single correlation between the continuous covariates: when p > 1, every off-diagonal entry of their correlation matrix is set to rho. It must lie

	in $(-1/(p-1), 1)$, outside of which the implied equicorrelation matrix is not positive definite; it is ignored when `p <= 1`.
`s`	Numeric. Standard deviation(s) of the continuous covariates. Either a single positive value shared by all `p` continuous covariates, or a vector of length `p` giving one positive standard deviation per covariate.
`sigma_t`	Numeric. Positive family scale/dispersion parameter σ_t. For `dist = "gamma"`, it is the conditional coefficient of variation. For the other families, it is the AFT scale parameter on the log-time distribution.
`mu_t`	Numeric. Intercept β_{t0} in the latent-time linear predictor. For `dist = "gamma"`, it is the log conditional mean when all covariates are zero. It is prepended to `beta_t` when forming the full coefficient vector.
`beta_t`	Numeric vector. The coefficients β_t for the AFT model. Combined with `mu_t`, the linear predictor is `cbind(1, x_i) %*% c(mu_t, beta_t)`.
`beta_g`	Numeric vector. The coefficients β_g for the prevalence model. The intercept β_{g0} is derived from `theta`.
`theta`	Numeric. Baseline prevalence parameter on the probability scale. Under: • `sel_mod = "probit"`: $\beta_{g0} = \mathrm{qnorm}(\theta)$. • `sel_mod = "logit"`: $\beta_{g0} = \log(\theta/(1-\theta))$.
`v_min`	Numeric. Minimum spacing for irregular screening intervals.
`v_max`	Numeric. Maximum spacing for irregular screening intervals.
`mean_rc`	Numeric. Mean of the exponential distribution controlling a random right-censoring time t_{rc} after the first screening.
`dist`	Character. Distribution for latent times t_i: `"weibull"`, `"lognormal"`, `"loglog"` (log-logistic), `"gamma"`, or `"gengamma"` (Prentice generalized gamma).
`q`	Numeric. Signed Prentice shape parameter for `dist = "gengamma"`; ignored otherwise.
`sel_mod`	Character. Either `"probit"` or `"logit"`, specifying the link function for the prevalence model.
`prob_r`	Numeric. Probability that a baseline test is performed ($r_i = 1$). If `prob_r = 0`, no baseline tests are done.

Details

The data-generating process includes:

1. **Covariates X:** Continuous multivariate normal distributed covariates are simulated using a correlation structure specified by `rho` and a common standard deviation `s`. If `p_discrete = 1`, a single discrete covariate is added, drawn from $\mathrm{Bernoulli}(0.5)$.
2. **Latent Times t:** An accelerated failure time (AFT) model is used, with linear predictor

$$\eta_i = \beta_{t0} + \beta_t^\top x_{ti},$$

where β_{t0} is set by `mu_t`. Covariates multiply event times by $\exp(\eta_i)$ relative to the corresponding baseline family.

```
For \code{"weibull"}, \code{"lognormal"}, and \code{"loglog"}
(log-logistic), latent times are generated from
\deqn{\log(t_i) = \eta_i + \sigma_t \epsilon_i.}{
          log(t_i) = eta_i + sigma_t * epsilon_i.}
For \code{"gamma"},
\deqn{t_i \mid x_{ti} \sim
      \mathrm{Gamma}{\sigma_t^{-2},
      \sigma_t^{-2}\exp(-\eta_i)},}{
      t_i | x_ti ~ Gamma(shape = sigma_t^-2,
      rate = sigma_t^-2 * exp(-eta_i)),}
so that \eqn{E(t_i \mid x_{ti}) = \exp(\eta_i)} and
\eqn{\sigma_t} is the conditional coefficient of variation.
For \code{"gengamma"}, the Prentice generalized gamma is generated
with location \eqn{\eta_i}, scale \eqn{\sigma_t}, and signed shape
parameter \code{q}.
```

3. **Irregular Screening Schedules V_i:** Each individual has multiple screening times generated randomly between `v_min` and `v_max`, ending in right censoring or the time of detection. These screening times (including a 0 for baseline and `Inf` for censoring) are returned in `v_obs`.
4. **Prevalence Indicator g_i:** Baseline prevalence is modeled via either a probit or logit link, consistent with:
$$w_i = \beta_{g0} + \beta_g^\top x_{gi} + \psi_i,$$
where β_{g0} is determined by `theta`, and β_g by `beta_g`. Specifically:
   - If `sel_mod = "probit"`, then $\beta_{g0} = \mathrm{qnorm}(\theta)$.
   - If `sel_mod = "logit"`, then $\beta_{g0} = \log(\theta/(1-\theta))$.

   We set $g_i = 1$ if $w_i > 0$, and $g_i = 0$ otherwise.
5. **Baseline Test Missingness r_i:** A baseline test indicator $r_i \in \{0, 1\}$ is drawn from a Bernoulli distribution with success probability `prob_r`, so $r_i = 1$ means the baseline test is performed and $r_i = 0$ means it is missing.
6. **Test Sensitivity κ:** A misclassification parameter κ (test sensitivity) can be specified via `kappa`. If $\kappa < 1$, some truly positive cases are missed.

Value

A list with the following elements:

`v_obs` A list of length `n`, each entry containing screening times. The first element is 0 (baseline), and `Inf` may indicate right censoring. The right-censoring time is drawn as the first post-baseline screening time plus an exponential increment, so it always exceeds that screening time and every individual receives at least one screening after baseline. Consequently the coding `c(0, Inf)`, a negative or missing baseline test followed by right censoring before the first regular screening, is never generated here, even though `bayespim` accepts and models that case.

`times_true` Numeric vector of length `n` giving the true latent times t_i.

`x` Numeric matrix of dimension $n \times p$ (plus an extra column if `p_discrete = 1`) containing the covariates.

`g` Binary vector of length `n`, indicating whether an individual is truly positive at baseline ($g_i = 1$).

`r` Binary vector of length `n`, indicating whether the baseline test was performed ($r_i = 1$) or missing ($r_i = 0$). This is the vector passed to the `r` argument of `bayespim`.

`prob_g` Numeric vector of length `n` giving the true prevalence probabilities, $P(g_i = 1)$.

References

T. Klausch, B. I. Lissenberg-Witte, and V. M. H. Coupé (2026). "A Bayesian prevalence-incidence mixture model for screening outcomes with misclassification.", Statistics in Medicine, 45(8-9), e70433. doi:10.1002/sim.70433

Examples

```
# Generate a small dataset for testing
set.seed(2025)
sim_data <- gen_data(n = 20, p = 1, p_discrete = 1,
                     sigma_t = 0.5, mu_t = 2,
                     beta_t = c(0.2, 0.2), beta_g = c(0.5, -0.2),
                     theta = 0.2,
                     dist = "weibull", sel_mod = "probit")
names(sim_data)
```

---

get_ic *Compute Information Criteria for a Bayesian Prevalence-Incidence Mixture Model*

Description

Computes and returns information criteria for a fitted Bayesian prevalence-incidence mixture model, including the Widely Applicable Information Criterion 1 (WAIC-1), WAIC-2, and the Deviance Information Criterion (DIC). These criteria are commonly used for model comparison and evaluation in Bayesian analysis. See Gelman et al. (2014) for further details on these criteria.

Usage

```
get_ic(mod, samples = NULL, cores = NULL)
```

Arguments

`mod`	A fitted prevalence-incidence mixture model of class `bayespim`.
`samples`	The number of MCMC samples to use. By default, all draws available after the model's warm-up are used. If the model was fitted with `save_every` > 1, this means all stored post-warm-up draws.
`cores`	The number of cores for parallel processing using `foreach`. If `NULL` (default), all available cores will be used.

Details

This function calculates information criteria for a fitted Bayesian prevalence-incidence mixture model (`bayespim`). The information criteria include:

- **WAIC-1**: Based on the sum of posterior variances of log-likelihood contributions.
- **WAIC-2**: Similar to WAIC-1 but incorporates an alternative variance estimate.
- **DIC**: Measures model fit by penalizing complexity via the effective number of parameters.

The computation is performed by evaluating log-likelihood values for MCMC samples. By default, all MCMC samples after burn-in are used, though a subset can be specified via the `samples` argument. All incidence-time distributions supported by `bayespim()` (Weibull, log-logistic, log-normal, gamma, and generalized gamma) are handled. For a model fitted with covariate standardization, likelihood calculations reconstruct the standardized parameterization and operate on standardized covariates to reduce cancellation on large original scales; this is algebraically equivalent to using the returned original-scale coefficients.

Parallelization is available via the `foreach` package, utilizing multiple cores if `cores` is set accordingly. If `cores = NULL`, all available cores will be used.

Value

A `matrix` containing WAIC-1, WAIC-2, and DIC values for the model.

References

Gelman, A., Hwang, J., & Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. Stat Comput, 24(6), 997–1016.

Examples

```
data(mod)
set.seed(2025)
get_ic(mod, samples = 20, cores = 1)
```

`log_aft_prior` *Log prior for accelerated failure time models*

Description

Evaluates the log-prior density of the incidence-model parameters in an accelerated failure time (AFT) model. For the Weibull, lognormal, log-logistic, and gamma distributions, the parameter vector is

Usage

```
log_aft_prior(eta, dist, beta_prior, tau_t, sig_prior, q_prior_sd)
```

Arguments

`eta`	Numeric AFT parameter vector. It contains the intercept and slope coefficients followed by `log(sigma)`. For `dist = "gengamma"`, the untransformed signed shape parameter `Q` is the final element.
`dist`	Character string identifying the AFT distribution. Supported values are `"weibull"`, `"lognormal"`, `"loglog"`, `"gamma"`, and `"gengamma"`.
`beta_prior`	Character string specifying the prior family for the regression coefficients. Supported values are `"norm"` and `"t"`.
`tau_t`	Numeric prior parameter. For `beta_prior = "t"`, this is the degrees of freedom of the Student-t prior. For `beta_prior = "norm"`, this is the standard deviation of the normal prior.
`sig_prior`	Positive numeric standard deviation of the half-normal prior on σ. For `dist = "gamma"`, σ is the conditional coefficient of variation.
`q_prior_sd`	Positive numeric standard deviation of the zero-centered normal prior on Q. It is ignored unless `dist = "gengamma"`.

Details

$$\eta = (\beta_0, \beta_1, \ldots, \beta_p, \log(\sigma)).$$

For the Prentice generalized-gamma distribution, the signed shape parameter Q is appended without transformation:

$$\eta = (\beta_0, \beta_1, \ldots, \beta_p, \log(\sigma), Q).$$

The same prior-function contract is used for every supported distribution and incidence sampler. A custom function supplied through `bayespim()` must accept the named arguments `eta`, `dist`, `beta_prior`, `tau_t`, `sig_prior`, and `q_prior_sd`, and must return one numeric log-density value. Arguments that are irrelevant to a custom prior may be ignored. When `bayespim()` is called with `standardize_covariates = TRUE`, `eta` contains the internally standardized incidence coefficients; returned posterior draws are transformed back to the original covariate scale only after sampling.

The default prior places either independent Student-t priors or independent zero-centered normal priors on the regression coefficients. A zero-centered half-normal prior is placed on the positive family scale/dispersion parameter σ. For the gamma model, σ is the conditional coefficient of variation; for the other families it is their AFT scale parameter. The log density includes the Jacobian for the transformation from σ to $\log(\sigma)$. For the generalized-gamma model, Q additionally receives a zero-centered normal prior with standard deviation `q_prior_sd`.

Value

A single numeric value giving the log-prior density.

Examples

```
# Use the default prior but give the incidence intercept five times its
# default prior scale. The function retains the complete prior contract and
# therefore works for all supported AFT distributions.
log_aft_prior_relaxed_intercept <- function(
    eta, dist, beta_prior, tau_t, sig_prior, q_prior_sd
) {
  log_prior <- log_aft_prior(
    eta = eta,
    dist = dist,
    beta_prior = beta_prior,
    tau_t = tau_t,
    sig_prior = sig_prior,
    q_prior_sd = q_prior_sd
  )

  if (beta_prior == "t") {
    log_prior -
      dt(eta[1], df = tau_t, log = TRUE) +
      dt(eta[1] / 5, df = tau_t, log = TRUE) -
      log(5)
  } else {
    log_prior -
      dnorm(eta[1], sd = tau_t, log = TRUE) +
      dnorm(eta[1], sd = 5 * tau_t, log = TRUE)
  }
}
```

---

mod *Converged example BayesPIM fit*

Description

A converged Weibull prevalence-incidence mixture model fitted to simulated screening data. The object is provided so examples of post-estimation functions can run without refitting the model.

Usage

```
data(mod)
```

Format

A `"bayespim"` object containing four MCMC chains with 2,000 generated and stored draws each (`save_every = 1`). The first 1,000 draws per chain are marked as warm-up. Its continuous covariate was standardized internally and its binary covariate was left unchanged; stored coefficients are on the original covariate scale. The fit contains the simulated screening observations, incidence and prevalence covariates, fitted parameter chains, convergence diagnostics, and the settings needed by BayesPIM post-estimation methods.

Details

The data were generated with `set.seed(2025)` using the data-generating settings shown in the main `bayespim()` documentation: 1,000 individuals, Weibull incidence times, two covariates, prevalence probability 0.2 at zero-valued covariates, and fixed test sensitivity 0.7.

The collapsed slice sampler was run once with automatic convergence updating enabled, although no update was required. All monitored parameters met R-hat at most 1.01 and effective sample size at least 400 after the initial 2,000 draws per chain.

Source

Simulated and fitted with `gen_data()` and `bayespim()`.

Examples

```
data(mod)
inherits(mod, "bayespim")
mod$convergence$converged
```

---

plot.bayespim *Plot method for bayespim objects*

Description

Plot method for bayespim objects

Usage

```
## S3 method for class 'bayespim'
plot(x, warmup = x$warmup, thinning = 1L, ...)
```

Arguments

`x`	An object of class `"bayespim"`.
`warmup`	Number of initial generated iterations to discard from each chain before optional plot-only thinning. Defaults to the final warmup value stored by `bayespim()`, including any update or explicit override. Supplying this argument overrides the stored value for this plot only.
`thinning`	Positive integer thinning interval used only for this plot. The default is no additional thinning. When the argument is omitted and more than 20,000 post-warmup draws are stored per chain, it is increased automatically to limit plotting cost.
`...`	Additional arguments passed to the MCMC plotting method.

Value

Invisibly returns `x`. The method is called for its side effect of producing trace and density plots for the requested parameter blocks.

Examples

```
data(mod)
plot(mod, thinning = 20)
```

`plot.ppCIF` *Plot posterior predictive cumulative incidence functions*

Description

Plot the mixture CIF, the non-prevalent CIF, or both CIFs from an object returned by `ppCIF`.

Usage

```
## S3 method for class 'ppCIF'
plot(
  x,
  y = NULL,
  type = c("mixture", "nonprevalent", "both"),
  ci = TRUE,
  main = NULL,
  xlab = "Time",
  ylab = "Cumulative incidence",
  xlim = NULL,
  ylim = c(0, 1),
  col = "#0072B2",
  ci_col = NULL,
  lwd = 2,
  ...
)
```

Arguments

`x`	An object of class `"ppCIF"`.
`y`	Ignored.
`type`	Character string selecting the CIF to draw: `"mixture"` (the default), `"nonprevalent"`, or `"both"`. The last choice produces horizontally adjacent mixture and non-prevalent panels on a common time scale.
`ci`	Logical. If `TRUE`, show pointwise 95 percent credible regions.
`main`	Optional plot title. For `type = "both"`, this may be a character vector of length two.
`xlab, ylab`	Axis labels.
`xlim, ylim`	Optional increasing numeric vectors of length two giving axis limits. A common `xlim` is used for both panels when `type = "both"`. The default probability limits are `c(0, 1)`.

col	Line color.
ci_col	Credible-region fill color. If NULL, a transparent version of col is used.
lwd	Line width.
...	Further graphical arguments passed to plot.default.

Details

The plotting contract is independent of the fitted incidence distribution and therefore also applies to generalized-gamma posterior predictions.

For ppd_type = "percentiles", time is shown on the x-axis and uncertainty is represented vertically. For ppd_type = "quantiles", the stored inverse-CDF representation is drawn with uncertainty in time. Pointwise regions can be omitted with ci = FALSE.

Value

Invisibly returns the supplied "ppCIF" object. The method is called for its base-graphics plotting side effect.

Examples

```
data(mod)
set.seed(2025)
cif <- ppCIF(
  mod,
  pst_samples = 50,
  quant = seq(0, 300, length.out = 51)
)
plot(cif, type = "both")
```

---

ppCIF *Posterior predictive cumulative incidence functions*

Description

Compute posterior predictive cumulative incidence functions (CIFs) from a fitted bayespim model.

Usage

```
ppCIF(
  mod,
  fix_x_t = NULL,
  fix_x_g = NULL,
  pst_samples = 1000,
  perc = seq(0, 1, 0.01),
  ppd_type = c("percentiles", "quantiles"),
  quant = NULL
)
```

Arguments

mod	A fitted model returned by bayespim(); objects without class "bayespim" are rejected.
fix_x_t	Either NULL for a marginal CIF or a numeric vector with one entry per incidence-model covariate. Numeric entries fix covariates at the supplied values; NA entries are integrated over the empirical covariate distribution.
fix_x_g	The corresponding vector for prevalence-model covariates. It cannot be supplied for a model fitted with prev = FALSE. When it is NULL, fix_x_t is also applied to the prevalence model if the two design matrices have the same number of columns.
pst_samples	Positive integer giving the number of posterior draws used for prediction. It cannot exceed the number of stored post-warm-up draws.
perc	Numeric vector of cumulative probabilities in $[0, 1]$ for which event times are returned when ppd_type = "quantiles".
ppd_type	Character string selecting the returned representation. The default, "percentiles", returns cumulative probabilities at the times in quant. "quantiles" returns event times at the cumulative probabilities in perc.
quant	Numeric vector of non-negative time points at which cumulative probabilities are evaluated when ppd_type = "percentiles". If NULL, a grid from zero to the maximum observed follow-up is used.

Details

A prevalence-incidence mixture model defines two related CIFs. One ppCIF() call always computes and stores both; the type argument of plot.ppCIF() controls which one is displayed. The mixture CIF includes prevalent cases as a point mass at time zero. The nonprevalent CIF describes event times among individuals who are non-prevalent at baseline. Both are computed from the same posterior draws and posterior predictive replicates and retained in the returned object. The candidate draws comprise every stored post-warm-up draw; no additional thinning is applied before the explicit random subsample of pst_samples draws. Fixed covariate values are supplied on their original scale. For a model fitted with covariate standardization, linear predictors are evaluated on the standardized computational scale to reduce numerical cancellation.

For a model fitted with prev = FALSE, the two CIFs are identical. Posterior prediction supports every incidence distribution available in bayespim(), including the Prentice generalized gamma. For dist = "gengamma", event times are drawn using the fitted location $\mu = \mathbf{x}_t^\top \boldsymbol{\beta}_t$, positive scale σ_t, and signed shape Q.

Value

An object of class "ppCIF" with:

mixture, nonprevalent Lists containing med_cdf, the pointwise posterior predictive median, and med_cdf_ci, a two-row matrix with the pointwise 2.5 and 97.5 percent posterior predictive quantiles.

ppd_type The selected representation.

quant, perc The applicable time or probability grid; the unused grid is NULL.

`pst_samples` The number of posterior draws used.

`distribution` The fitted incidence distribution.

`prevalence_model` Whether prevalence was fitted.

`call` The matched call.

See Also

`plot.ppCIF`, `bayespim`

Examples

```
data(mod)
set.seed(2025)
cif <- ppCIF(
  mod,
  pst_samples = 50,
  ppd_type = "percentiles",
  quant = seq(0, 300, length.out = 51)
)
plot(cif)
plot(cif, type = "nonprevalent")
plot(cif, type = "both")
```

---

search_prop_sd *Automated Heuristic Search of a Proposal Standard Deviation for* `bayespim`

Description

When bayespim is fitted with `sampler = "mh"`, it uses a Metropolis step for sampling the incidence-model parameters and requires a standard deviation for the normal proposal distribution. This function uses a heuristic algorithm to find a proposal standard deviation such that the Metropolis sampler accepts proposed draws at a rate within the user-defined interval (by default around 20–25%). The default sampler in `bayespim()` is `"slice_collapsed"`, so the Metropolis sampler must be requested explicitly before using this function.

Usage

```
search_prop_sd(m, ndraws = 1000, succ_min = 3, acc_bounds = c(0.2, 0.25))
```

Arguments

`m`	A model object of class `bayespim`.
`ndraws`	Starting number of MCMC iterations after which the acceptance rate is first evaluated. Defaults to 1000.

succ_min The algorithm doubles the number of MCMC draws succ_min times (each time the acceptance rate is within acc_bounds), ensuring stability. Defaults to 3.

acc_bounds A numeric vector of length two specifying the lower and upper bounds for the acceptable acceptance rate. Defaults to c(0.2, 0.25).

Details

Starting from an initial bayespim model object m, the function attempts to calibrate the standard deviation of the proposal distribution. Specifically, it:

1. Runs an initial update of ndraws iterations and computes an acceptance rate.
2. If the acceptance rate lies within acc_bounds, the number of MCMC draws ndraws is doubled, and the process repeats.
3. Otherwise, the proposal standard deviation σ is adjusted based on whether the acceptance rate p is below the lower bound a or above the upper bound b of acc_bounds.
4. The formula for adjustment is:

$$\sigma \leftarrow \sigma \times (1 - \frac{(a-p)}{a}) \quad \text{if } p < a, \quad \sigma \leftarrow \sigma \times (1 + \frac{(p-b)}{b}) \quad \text{if } p > b.$$

By default, if the acceptance rate falls within $[0.2, 0.25]$, that σ is considered acceptable, and the process continues until succ_min consecutive successes (doubles) are achieved.

Value

A list with the following elements:

prop_sd The final (adjusted) proposal standard deviation.

ac The acceptance rate in the last iteration.

Examples

```
## Not run:
# search_prop_sd() requires an MH fit. This deliberately short initial fit
# illustrates the interface; use more draws for an actual calibration.
set.seed(2025)
dat <- gen_data(
  kappa = 0.7, n = 100, theta = 0.2,
  p = 1, p_discrete = 1,
  beta_t = c(0.2, 0.2), beta_g = c(0.2, 0.2),
  v_min = 20, v_max = 30, mean_rc = 80,
  sigma_t = 0.2, mu_t = 5, dist = "weibull",
  prob_r = 1
)

fit_mh <- bayespim(
  v_obs = dat$v_obs, x_t = dat$x, x_g = dat$x, r = dat$r,
  kappa = 0.7, update_kappa = FALSE,
  ndraws = 100, warmup = 10, chains = 2,
  seed_chains = c(202501, 202502),
```

```
  update_till_converge = FALSE,
  sampler = "mh", prop_sd = 0.005, dist = "weibull",
  silent = TRUE
)

search_sd <- search_prop_sd(
  m = fit_mh,
  ndraws = 100,
  succ_min = 1
)
print(search_sd)

## End(Not run)
```

`summary.bayespim` *Summary method for bayespim objects*

Description

Summary method for bayespim objects

Usage

```
## S3 method for class 'bayespim'
summary(object, warmup = object$warmup, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

`object`	An object of class `"bayespim"`.
`warmup`	Number of initial generated MCMC iterations to discard from each chain before calculating the summary from every stored draw. It is interpreted on the generated-iteration scale, not as a number of stored draws when `save_every` > 1. Defaults to the final warmup value stored by `bayespim()`, including any update or explicit override. Supplying this argument overrides the stored value for this summary only.
`probs`	Numeric vector of posterior quantiles.
`...`	Additional arguments, currently unused.

Value

Invisibly returns a list of class `"summary.bayespim"` containing the fitted latent-time distribution and incidence sampler, posterior and convergence tables for the latent-time model and prevalence model, and kappa when estimated, together with the convergence criteria, formatted draw information, and the underlying draw counts. The same information is printed to the console.

Examples

```
data(mod)
summary(mod)
```

`trim_mcmc` *Trim and thin an* `mcmc.list`

Description

Convenience function for trimming burn-in iterations and applying thinning to an object of class `mcmc.list`.

Usage

```
trim_mcmc(obj, burnin = 0L, end = NULL, thinning = 1L)
```

Arguments

`obj`	An object of class `mcmc.list`.
`burnin`	Non-negative integer giving the number of stored rows to discard. Defaults to `0`.
`end`	Integer; final stored row to retain. Defaults to the number of rows in the first chain.
`thinning`	Integer; thinning interval. Defaults to `1`.

Details

The function selects stored rows `seq(burnin + 1, end, by = thinning)` from each chain and reconstructs the result as an `mcmc.list`, preserving its original iteration numbering and multiplying its existing storage interval by `thinning`.

Value

An object of class `mcmc.list` containing the trimmed and thinned chains.

Examples

```
data(mod)
trimmed <- trim_mcmc(
  mod$par,
  burnin = mod$warmup,
  thinning = 20
)
nrow(as.matrix(trimmed[[1]]))
```

Index

∗ **datasets**
mod, 17

bayespim, 2, *13, 14*, *22*
bayespim(), *15, 16*

gen_data, 11
get_ic, 14

log_aft_prior, 15

mod, 17

plot.bayespim, 18
plot.bayespim(), *10*
plot.default, *20*
plot.ppCIF, 19, *22*
ppCIF, *19*, 20

search_prop_sd, *4*, *7*, 22
summary.bayespim, 24
summary.bayespim(), *10*

trim_mcmc, 25