Package 'BayesPET'

February 12, 2026

Title Bayesian Prediction of Event Times for Blinded Randomized Controlled Trials

Version 0.1.0

Date 2026-02-04

Description Bayesian methods for predicting the calendar time at which a target number of events is reached in clinical trials. The methodology applies to both blinded and unblinded settings and jointly models enrollment, event-time, and censoring processes. The package provides tools for trial data simulation, model fitting using 'Stan' via the 'rstan' interface, and event time prediction under a wide range of trial designs, including varying sample sizes, enrollment patterns, treatment effects, and event or censoring time distributions. The package is intended to support interim monitoring, operational planning, and decision-making in clinical trial development. Methods are described in Fu et al. (2025) <doi:10.1002/sim.70310>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Biarch true

Depends R (>= 4.1.0)

Imports dplyr, magrittr, methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), utils, furrr, future, readr, tibble, tidyr, reshape2

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements GNU make

LazyData true

NeedsCompilation yes

Author Xinyi He [cre, aut], Jingyan Fu [aut], Ying Yuan [aut, cph]

Maintainer Xinyi He <xinyi.he@uth.tmc.edu>

Repository CRAN

Date/Publication 2026-02-12 08:00:26 UTC

Contents

BayesPET-package . . . 2
convert_median . . . 3
data_example . . . 6
fit_censor . . . 8
fit_enroll . . . 10
fit_event_blind . . . 13
fit_event_unblind . . . 16
fit_models . . . 19
generate_data . . . 23
get_oc . . . 27
plot.BayesPET_predtime . . . 32
predict_eventtime . . . 33
print.BayesPET_fit . . . 37
print.BayesPET_predtime . . . 38
summary.BayesPET_oc . . . 39
summary.BayesPET_predtime . . . 41

Index **43**

`BayesPET-package` *BayesPET: Bayesian Prediction of Event Times for Blinded Randomized Controlled Trials*

Description

The package implements Bayesian models for predicting the calendar time at which a target number of events is reached in clinical trials. The methodology applies to both blinded and unblinded settings and jointly models enrollment, event-time, and censoring processes.

Details

The package provides tools for trial data simulation, model fitting using 'Stan' via the 'rstan' interface, and event time prediction under a wide range of trial designs, including varying sample sizes, enrollment patterns, treatment effects, and event or censoring time distributions. The package is intended to support interim monitoring, operational planning, and decision-making in clinical trial development. For details, please refer to Fu et al. (2025) doi:10.1002/sim.70310.

Author(s)

Xinyi He <xinyi.he@uth.tmc.edu>, Jingyan Fu, and Ying Yuan

References

Fu, J., Zhao, D., Skanji, D., Liu, H., Tang, R. S., and Yuan, Y. (2025). *Bayesian Prediction of Event Times Using Mixture Model for Blinded Randomized Controlled Trials*. *Statistics in Medicine*, 44(28–30), e70310. doi:10.1002/sim.70310

Stan Development Team (2023). *RStan: The R Interface to Stan*. R package version 2.32.7. https://mc-stan.org

---

`convert_median` *Solve baseline survival parameters by matching the marginal median survival time*

Description

In practice, it is sometimes more convenient to work with a marginal (population-level) median survival time. This function solves for exactly one unknown baseline quantity - `shape`, `scale`, or `median` - in a proportional hazards (PH) survival model with either a Weibull or log-logistic baseline distribution, given the other two.

When covariates are provided, `median` is interpreted as the marginal (population-level) median survival time defined implicitly by

$$\mathbb{E}_{\boldsymbol{Z}}\{S(\text{median} \mid \boldsymbol{Z})\} = 0.5$$

and the unknown quantity is obtained by Monte Carlo integration over the covariate distribution.

Usage

```
convert_median(
  distribution = "Weibull",
  shape,
  median,
  scale,
  cov_type = NULL,
  cov_dist = NULL,
  beta = NULL,
  S = 20000,
  seed.solveparam = 123,
  interval = if (is.null(scale)) c(1e-15, 100) else c(1e-06, 20000),
  tol = .Machine$double.eps^0.25,
  maxiter = 1000
)
```

Arguments

`distribution`	Character string specifying the baseline survival distribution. Must be either `"Weibull"` or `"Loglogistic"` (log-logistic). Defaults to `"Weibull"`.

`shape` Positive numeric scalar. Baseline shape parameter. For `distribution = "Weibull"`, this corresponds to the Weibull shape ρ. For `distribution = "Loglogistic"`, this corresponds to the log-logistic shape a. Set to `NULL` to solve for the shape.

`median` Positive numeric scalar. Median survival time. If covariates are provided, this is the *marginal* (population) median m defined by

$$\mathbb{E}_{\boldsymbol{Z}}\{S(m \mid \boldsymbol{Z})\} = 0.5,$$

where $\boldsymbol{Z}$ denotes the covariate vector and the expectation is taken with respect to the covariate distribution implied by `cov_type` and `cov_dist`. Set to `NULL` to solve for `median`.

`scale` Positive numeric scalar. Baseline scale parameter. For `"Weibull"`, this is the Weibull scale parameter λ_0. For `"Loglogistic"`, this is the log-logistic scale b. Set to `NULL` to solve for the scale.

`cov_type` Character vector specifying the distribution for each component of the covariate vector $\boldsymbol{Z} = (Z_1, \ldots, Z_p)$ used in the proportional hazards model. Each element must be `"binary"` or `"continuous"`. If `NULL` (default), no covariates are used.

`cov_dist` Numeric vector of the same length as `cov_type`, giving parameters for the covariate-generating distribution of each Z_j:

- `"binary"`: $Z_j \sim \text{Bernoulli}(p_j)$ with $p_j = cov\_dist[j]$.
- `"continuous"`: $Z_j \sim N(0, \sigma_j^2)$ with $\sigma_j = cov\_dist[j]$.

If `NULL` (default), no covariates are used.

`beta` Numeric vector of regression coefficients $\boldsymbol{\beta}$ in the proportional hazards linear predictor $\boldsymbol{Z}^\top\boldsymbol{\beta}$. Must have the same length as `cov_type`. If `NULL` (default), no covariates are used.

`S` Integer. Monte Carlo sample size used to approximate the marginal survival when covariates are provided. Defaults to `20000`.

`seed.solveparam` Integer random seed for covariate simulation; if `NULL`, the RNG state is not reset. Defaults to `123`.

`interval` Numeric vector of length 2 giving the lower and upper bounds passed to `uniroot` for the root-finding procedure used to solve for the unknown parameter (shape, scale, or median). The interval must bracket the true solution, i.e., the function values at the two endpoints must have opposite signs. Defaults to `c(1e-15, 100)` when `scale = NULL` and `c(1e-6, 20000)` otherwise.

`tol` Numeric scalar. Defaults to `.Machine$double.eps^0.25`. Convergence tolerance passed to `uniroot`.

`maxiter` Integer giving the maximum number of iterations allowed for `uniroot`. An error is raised if the algorithm fails to converge within this limit. Defaults to `1000`.

Details

The proportional hazards (PH) model assumes that, conditional on a covariate vector $\boldsymbol{Z}$, the hazard function satisfies

$$h(t \mid \boldsymbol{Z}) = h_0(t)\exp(\boldsymbol{Z}^\top\boldsymbol{\beta})$$

for $t \geq 0$, where $h_0(t)$ is the baseline hazard and $\boldsymbol{\beta}$ is a vector of regression coefficients.

The baseline model is specified by `distribution` through the baseline survival function $S_0(t)$:

- `"Weibull"`: baseline survival is

$$S_0(t) = \exp\{-\lambda_0 t^\rho\}$$

for $t \geq 0$, where $\rho > 0$ is the `shape` parameter and $\lambda_0 > 0$ is the `scale` parameter. An equivalent representation used by several standard implementations is

$$S_0(t) = \exp\{-(t/\sigma_0)^\rho\}$$

for $t \geq 0$, where $\sigma_0 > 0$ is a reparameterization satisfying $\lambda_0 = \sigma_0^{-\rho}$. This latter form is used by `dweibull`, `phreg`, `flexsurvreg`, and `rstan`.

- `"Loglogistic"`: baseline survival is

$$S_0(t) = \{1 + (t/b)^a\}^{-1}$$

for $t \geq 0$, where $a > 0$ is the `shape` parameter and $b > 0$ is the `scale` parameter.

Under the PH assumption, the conditional survival satisfies

$$S(t \mid \boldsymbol{Z}) = S_0(t)^{\exp(\boldsymbol{Z}^\top \boldsymbol{\beta})}.$$

When no covariates are supplied (`cov_type = NULL`), the median survival time corresponds to the baseline median and closed-form solutions are available for some distributions.

When covariates are supplied, the median survival time m is interpreted as the marginal (population) median defined implicitly by

$$\mathbb{E}_{\boldsymbol{Z}}\{S(m \mid \boldsymbol{Z})\} = 0.5,$$

where the expectation is taken with respect to the covariate distribution implied by `cov_type` and `cov_dist`. This expectation is approximated using Monte Carlo simulation with `S` draws, and the unknown parameter is obtained by numerical root finding via `uniroot`.

Value

A numeric scalar giving the solved parameter:

- the `shape` parameter if `shape = NULL`;
- the baseline `scale` parameter if `scale = NULL`;
- the `median` survival time if `median = NULL`.

See Also

`dweibull` `rllogis`

Examples

```
# Weibull: convert a desired median to the corresponding scale parameter
## No covariates
convert_median(
  distribution = "Weibull",
  shape = 3,
  median = 5,
  scale = NULL,
  seed = 123
)
## With covariates
convert_median(
  distribution = "Weibull",
  shape = 3,
  median = 5,
  scale = NULL,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, 1),
  beta = c(1, 0.5),
  seed = 123
)

# Log-logistic: convert median to scale when covariates enter the model
convert_median(
  distribution = "Loglogistic",
  shape = 1.5,
  median = 3,
  scale = NULL,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, 1),
  beta = c(1, 0.5),
  seed = 123
)
```

`data_example` *Example trial datasets for fitting Stan models and predicting event times*

Description

`data_example` is a simulated set of trial datasets designed to illustrate a complete analysis workflow, including fitting enrollment and survival and censoring models using interim data, and predicting the calendar time at which a prespecified number of events is expected to be observed.

Context. Consider a two-arm time-to-event trial that plans to enroll $N = 200$ participants, with equal randomization between the experimental and control arms. An interim analysis is conducted after $E_{\text{cutoff}} = 75$ events have occurred, and the final analysis is planned at $E_{\text{target}} = 150$ events. Based on the data available at the interim analysis, the goal is to predict the calendar time (measured from the trial start) at which the cumulative number of events will reach E_{target}.

The bundle contains:

`example_enroll` Enrollment information (interarrival times and enrollment status).

`example_eventcensor` Treatment assignment, follow-up time, event indicator, censoring indicator, and covariates.

Usage

```
data(data_example)
```

Format

A named list with two data frames:

`example_enroll` A data frame with columns:

No Integer subject identifier.

interarrivaltime Interarrival time.

enrollstatus Enrollment status indicator (`1` = enrolled, `0` = administratively censored).

`example_eventcensor` A data frame with columns:

No Integer subject identifier.

trt Treatment assignment indicator (`1` = experimental, `0` = control). This column is used for unblinded analyses and can be ignored or set to `NA` when fitting blinded event time models.

time Observed follow-up time (event or censoring time, positive).

eventstatus Event indicator (`1` = event, `0` = right-censored).

censorstatus Random censoring indicator (`1` = random censoring observed, `0` = no random censoring, including administrative censoring or observed event).

X1 Binary covariate (0/1).

X2 Continuous covariate.

Source

Simulated data.

Examples

```
data(data_example)
names(data_example)
head(data_example$example_enroll)
head(data_example$example_eventcensor)
```

`fit_censor` *Fit a Weibull model for random censoring times*

Description

Fits a Weibull proportional hazards model for random censoring times and returns posterior draws of the Weibull shape parameter $\rho_c > 0$, the baseline scale parameter $\lambda_{0c} > 0$, and, when covariates are included, the covariate log hazard ratios $\boldsymbol{\beta}_c$.

Usage

```
fit_censor(
  t_obs,
  status_censor,
  cov = NULL,
  hyperparams_censor = list(),
  chains = 4,
  iter = 4000,
  seed = 2,
  refresh = 0,
  mc.cores = 1,
  warmup = floor(iter/2),
  control = list(adapt_delta = 0.95),
  return_fit = FALSE,
  quiet = TRUE
)
```

Arguments

`t_obs`	Numeric vector of observed times t^*_{ci}. Must be `>0`.
`status_censor`	Integer vector of random censoring indicators δ_{ci} with values `0` or `1`. A value of `1` indicates the random censoring time is observed, while `0` indicates administrative censoring or event at `t_obs`.
`cov`	Optional matrix or data frame of covariates $\boldsymbol{Z}$. Each row corresponds to a subject and each column to a covariate. Covariates must be numeric and are treated as linear effects in the model. Only binary (0/1) and continuous covariates are supported. Multilevel or categorical variables are not expanded into dummy variables; if provided as numeric (e.g., factor codes), they are treated as continuous. If `NULL` (default), a no-covariate Weibull model is fitted.
`hyperparams_censor`	A named list of prior hyperparameters for the censoring model. If it is empty (`list()`), default values are assigned internally. • `alpha_c`, `beta_c`: Shape and rate parameters of the Gamma prior for the baseline scale parameter λ_{0c} (defaults: 0.1, 0.1). • `alpha_rc`, `beta_rc`: Shape and rate parameters of the Gamma prior for the baseline shape parameter ρ_c (defaults: 0.1, 0.1).

- `mu_bc, sigma_bc`: Mean and standard deviation of the Normal prior for the covariate effects β_{cj} (defaults: 0, $\sqrt{10}$).

No other hyperparameters are allowed.

`chains` Number of Markov chain Monte Carlo (MCMC) chains. Defaults to `4`.

`iter` Number of iterations per chain (including warmup). Defaults to `4000`.

`seed` Optional integer seed passed to `sampling` for reproducibility. If `NULL`, 'Stan' generates a seed internally and results may differ across runs. Defaults to `2`.

`refresh` Frequency of progress updates from `sampling`. Set to `0` (default) to suppress output.

`mc.cores` Integer. Number of CPU cores to use when executing Markov chains in parallel via `sampling`. Defaults to `1`. We recommend setting `mc.cores` to the maximum number of processors supported by the available hardware and memory, up to the number of chains.

`warmup` Number of warmup (burn-in) iterations per chain. Must be strictly smaller than `iter`. Defaults to `floor(iter / 2)`.

`control` A named list of sampler control parameters passed to `sampling`. Examples include `adapt_delta` and `max_treedepth`. Defaults to `list(adapt_delta = 0.95)`.

`return_fit` Logical; if `TRUE`, also return the underlying 'rstan' `stanfit` object. Defaults to `FALSE`.

`quiet` Logical. If `TRUE` (default), suppress messages and diagnostic warnings from 'Stan' during model fitting. Useful for large simulation studies.

Details

Let T_{ci} denote the censoring time for subject i. Let $\boldsymbol{Z}_i$ denote the covariate vector for subject i (the i-th row of `cov` when provided). The parameterization has baseline density function

$$f_{0c}(t) = \lambda_{0c}\,\rho_c\,t^{\rho_c-1}\exp(-\lambda_{0c}t^{\rho_c}), \quad t \geq 0.$$

The model assumes the survival function

$$S_c(t \mid \boldsymbol{Z}_i) = \exp\{-\lambda_{0c}\,t^{\rho_c}\exp(\boldsymbol{Z}_i^\top\boldsymbol{\beta}_c)\}, \quad t \geq 0.$$

The corresponding hazard function is

$$h_c(t \mid \boldsymbol{Z}_i) = \lambda_{0c}\,\rho_c\,t^{\rho_c-1}\exp(\boldsymbol{Z}_i^\top\boldsymbol{\beta}_c), \quad t \geq 0.$$

Note: In 'Stan', the Weibull distribution is parameterized by shape parameter ρ_c and scale parameter σ, with baseline density function

$$f_{0c}(t) = (\rho_c/\sigma)\,(t/\sigma)^{\rho_c-1}\,\exp(-(t/\sigma)^{\rho_c}), \quad t \geq 0.$$

To match the hazard-scale parameterization above, we set

$$\sigma_i = \{\lambda_{0c}\exp(\boldsymbol{Z}_i^\top\boldsymbol{\beta}_c)\}^{-1/\rho_c}.$$

The random censoring indicator `status_censor` follows the convention:

- `0`: random censoring time observed at `t_obs` (density contribution),
- `1`: administratively right-censored at `t_obs` (survival contribution).

Value

A list with the following components:

- `rho_c`: Posterior draws of the Weibull shape parameter ρ_c.
- `lambda_0c`: Posterior draws of the baseline Weibull scale parameter λ_{0c}.
- `beta_c`: Posterior draws of the covariate log hazard ratios $\boldsymbol{\beta}_c$, or `NULL` if no covariates are included.
- `fit`: The 'rstan' `stanfit` object (only if `return_fit = TRUE`).

See Also

Other BayesPET model fitting: `fit_enroll`, `fit_event_blind`, `fit_event_unblind`, `fit_models`, `print.BayesPET_fit`

Examples

```
data(data_example)
example_eventcensor<-data_example$example_eventcensor

## ---- fit censoring model ----
## Reduced number of chains and iterations compared to defaults
## to keep the example computationally manageable.
fit <- fit_censor(
  t_obs = example_eventcensor$time,
  status_censor = example_eventcensor$censorstatus,
  cov = example_eventcensor[,6:7],
  chains = 2,
  iter = 2000, quiet = FALSE,
  seed = 2, return_fit = TRUE
)

summary(fit$rho_c)
summary(fit$lambda_0c)
summary(fit$beta_c)
print(fit$fit)
```

`fit_enroll` *Fit enrollment model*

Description

Fits an exponential model to enrollment interarrival times and returns posterior draws of the enrollment rate μ.

Usage

```
fit_enroll(
  status_enroll,
  t_enroll,
  hyperparams_enroll = list(),
  chains = 4,
  iter = 4000,
  seed = 1,
  refresh = 0,
  warmup = floor(iter/2),
  mc.cores = 1,
  control = list(adapt_delta = 0.95),
  return_fit = FALSE,
  quiet = TRUE
)
```

Arguments

status_enroll	Integer vector with values 0 or 1 indicating whether the interarrival time is observed (1) or administratively censored (0).
t_enroll	Numeric vector giving observed or censored interarrival times. Must be non-negative and finite.
hyperparams_enroll	A named list of prior hyperparameters for the enrollment model. If it is empty (list()), default values are assigned internally. • alpha_mu: Shape parameter of the Gamma prior for the enrollment rate μ (default: 0.1). • beta_mu: Rate parameter of the Gamma prior for the enrollment rate μ (default: 0.1). No other hyperparameters are allowed.
chains	Number of Markov chain Monte Carlo (MCMC) chains. Defaults to 4.
iter	Number of iterations per chain (including warmup). Defaults to 4000.
seed	Optional integer seed passed to sampling for reproducibility. If NULL, 'Stan' generates a seed internally and results may differ across runs. Defaults to 1.
refresh	Frequency of progress updates from sampling. Set to 0 (default) to suppress output.
warmup	Number of warmup (burn-in) iterations per chain. Must be strictly smaller than iter. Defaults to floor(iter / 2).
mc.cores	Integer. Number of CPU cores to use when executing Markov chains in parallel via sampling. Defaults to 1. We recommend setting mc.cores to the maximum number of processors supported by the available hardware and memory, up to the number of chains.
control	A named list of sampler control parameters passed to sampling. Examples include adapt_delta and max_treedepth. Defaults to list(adapt_delta = 0.95).

return_fit	Logical; if TRUE, also return the underlying 'rstan' stanfit object. Defaults to FALSE.
quiet	Logical. If TRUE (default), suppress messages and diagnostic warnings from sampling during model fitting. Useful for large simulation studies.

Details

Let t_i denote the interarrival time between consecutive enrollments. The enrollment process is modeled using an exponential distribution with rate parameter μ, so that

$$t_i \sim \text{Exponential}(\mu),$$

where $\mu > 0$ represents the average enrollment rate. The corresponding density is

$$f(t) = \mu e^{-\mu t}, \quad t \geq 0.$$

Administrative censoring of enrollment is handled through the indicator status_enroll, which follows the convention:

- 1: the interarrival time is fully observed at t_enroll (density contribution);
- 0: the interarrival time is administratively censored at t_enroll (survival contribution).

Value

A list with components:

- mu: posterior draws of μ,
- fit: The 'rstan' stanfit fit object (only if return_fit = TRUE).

See Also

Other BayesPET model fitting: fit_censor, fit_event_blind, fit_event_unblind, fit_models, print.BayesPET_fit

Examples

```
data(data_example)
example_enroll<-data_example$example_enroll

out <- fit_enroll(
  status_enroll = example_enroll$enrollstatus,
  t_enroll = example_enroll$interarrivaltime,
  seed = 1, return_fit = TRUE,
  quiet = FALSE
)

summary(out$mu)
print(out$fit)
```

`fit_event_blind` *Fit a Weibull event-time model with unknown treatment assignments*

Description

Fits a Weibull event-time model formulated as a mixture over unobserved treatment assignments. The function returns posterior draws of the Weibull shape parameter ρ_e, the baseline scale parameter λ_{0e}, the treatment log hazard ratio η, and when covariates are included, the covariate log hazard ratios $\boldsymbol{\beta}_e$. Posterior draws of the latent treatment indicators x_i are also returned.

Usage

```
fit_event_blind(
  t_event,
  status_event,
  p_trt,
  cov = NULL,
  hyperparams_event = list(),
  chains = 4,
  iter = 4000,
  seed = 123,
  refresh = 0,
  warmup = floor(iter/2),
  mc.cores = 1,
  control = list(adapt_delta = 0.95),
  return_fit = FALSE,
  quiet = TRUE
)
```

Arguments

`t_event`	Numeric observed time to event t^*_{ei}. Must be `>0`.
`status_event`	Integer vector of event indicators δ_{ei} with values `0` or `1` (`1` = event observed, `0` = right-censored at `t_event`).
`p_trt`	Scalar randomization probability to the experimental arm, $\gamma \in (0, 1)$.
`cov`	Optional matrix or data frame of covariates $\boldsymbol{Z}$. Each row corresponds to a subject and each column to a covariate. Covariates must be numeric and are treated as linear effects in the model. Only binary (0/1) and continuous covariates are supported. Multilevel or categorical variables are not expanded into dummy variables; if provided as numeric (e.g., factor codes), they are treated as continuous. If `NULL` (default), a no-covariate Weibull model is fitted.
`hyperparams_event`	A named list of prior hyperparameters for the event-time model. If it is empty (`list()`), default values are assigned internally. • `alpha_e`, `beta_e`: Shape and rate parameters of the Gamma prior for the baseline scale parameter λ_{0e} (defaults: 0.1, 0.1).

- `alpha_re`, `beta_re`: Shape and rate parameters of the Gamma prior for the baseline shape parameter ρ_e (defaults: 0.1, 0.1).
- `mu_eta`, `sigma_eta`: Mean and standard deviation of the Normal prior, truncated to $[0, \infty)$, for the treatment effect (log hazard ratio) η (defaults: 0, $\sqrt{2}$).
- `mu_be`, `sigma_be`: Mean and standard deviation of the Normal prior for the covariate effects β_{ej} (defaults: 0, $\sqrt{10}$).

No other hyperparameters are allowed.

`chains` Number of Markov chain Monte Carlo (MCMC) chains. Defaults to `4`.

`iter` Number of iterations per chain (including warmup). Defaults to `4000`.

`seed` Optional integer seed passed to `sampling` for reproducibility. If `NULL`, 'Stan' generates a seed internally and results may differ across runs. Defaults to `123`.

`refresh` Frequency of progress updates from `sampling`. Set to `0` (default) to suppress output.

`warmup` Number of warmup (burn-in) iterations per chain. Must be strictly smaller than `iter`. Defaults to `floor(iter / 2)`.

`mc.cores` Integer. Number of CPU cores to use when executing Markov chains in parallel via `sampling`. Defaults to `1`. We recommend setting `mc.cores` to the maximum number of processors supported by the available hardware and memory, up to the number of chains.

`control` A named list of sampler control parameters passed to `sampling`. Examples include `adapt_delta` and `max_treedepth`. Defaults to `list(adapt_delta = 0.95)`.

`return_fit` Logical; if `TRUE`, also return the underlying 'rstan' `stanfit` object. Defaults to `FALSE`.

`quiet` Logical. If `TRUE` (default), suppress messages and diagnostic warnings from 'Stan' during model fitting. Useful for large simulation studies.

Details

Let T_{ei} denote the event time for subject i. Let $\boldsymbol{Z}_i$ denote the corresponding covariate vector, which is the i th row of `cov` when provided. Treatment assignment is represented by a latent indicator x_i, where $x_i = 0$ denotes control and $x_i = 1$ denotes experimental treatment. The latent treatment indicators are jointly inferred with the model parameters from the observed event time data.

The treatment effect parameter η represents the log hazard ratio comparing experimental treatment to control, and $\boldsymbol{\beta}_e$ denotes the vector of covariate regression coefficients (log hazard ratios) in the proportional hazards model.

The baseline event time distribution follows a Weibull model with density

$$f_{0e}(t) = \lambda_{0e}\, \rho_e\, t^{\rho_e - 1} \exp(-\lambda_{0e} t^{\rho_e}), \quad t \geq 0.$$

Conditional on treatment and covariates, the hazard function is

$$h_e(t \mid x_i, \boldsymbol{Z}_i) = \lambda_{0e}\, \rho_e\, t^{\rho_e - 1} \exp(\eta x_i + \boldsymbol{Z}_i^\top \boldsymbol{\beta}_e), \quad t \geq 0,$$

and the corresponding survival function is

$$S_e(t \mid x_i, \boldsymbol{Z}_i) = \exp\{-\lambda_{0e}\, t^{\rho_e} \exp(\eta x_i + \boldsymbol{Z}_i^\top \boldsymbol{\beta}_e)\}, \quad t \geq 0.$$

For reference, 'Stan' uses an equivalent Weibull representation based on a shape parameter ρ_e and a scale parameter σ, with baseline density

$$f_{0e}(t) = (\rho_e/\sigma)\,(t/\sigma)^{\rho_e - 1} \exp\{-(t/\sigma)^{\rho_e}\}, \quad t \geq 0.$$

The two formulations are related through

$$\sigma_i = \{\lambda_{0e} \exp(\eta x_i + \boldsymbol{Z}_i^\top \boldsymbol{\beta}_e)\}^{-1/\rho_e}.$$

Because x_i is not observed in blinded randomized trials, it is treated as a latent variable, and the observed event-time data marginally follow a mixture of two Weibull distributions corresponding to the latent treatment groups.

To avoid label switching in posterior inference, the treatment effect parameter η is assigned a Normal prior truncated to $[0, \infty)$, restricting η to be nonnegative.

Value

A list with the following components:

- `eta`: Posterior draws of the treatment log hazard ratio η.
- `rho_e`: Posterior draws of the Weibull shape parameter ρ_e.
- `lambda_0e`: Posterior draws of the baseline Weibull scale parameter λ_{0e}.
- `beta_e`: Posterior draws of the covariate log hazard ratios $\boldsymbol{\beta}_e$, or `NULL` if no covariates are included.
- `x`: Posterior draws of latent treatment indicators x_i.
- `fit`: The 'rstan' `stanfit` object (only if `return_fit = TRUE`).

See Also

Other BayesPET model fitting: `fit_censor`, `fit_enroll`, `fit_event_unblind`, `fit_models`, `print.BayesPET_fit`

Examples

```
data(data_example)
example_eventcensor<-data_example$example_eventcensor
# Use 2 chains and iter = 1000 here to reduce runtime for the example;
# use more chains in real analyses.
fit_e_blind <- fit_event_blind(
  t_event = example_eventcensor$time,
  status_event = example_eventcensor$eventstatus,
  cov = example_eventcensor[,6:7],
  p_trt = 0.5,
  chains = 2,
  iter = 1000, seed = 123,
```

```
  return_fit = TRUE
)

summary(fit_e_blind$eta)
print(fit_e_blind$fit)
```

`fit_event_unblind` *Fit a Weibull event-time model with known treatment assignments*

Description

Fits a Weibull event time model in which treatment assignments are observed. The function returns posterior draws of the Weibull shape parameter $\rho_e > 0$, the baseline scale parameter $\lambda_{0e} > 0$, the treatment effect coefficient (log hazard ratio) η, and when covariates are included, the covariate log hazard ratios $\boldsymbol{\beta}_e$.

Usage

```
fit_event_unblind(
  t_event,
  status_event,
  treatment_ind,
  cov = NULL,
  hyperparams_event = list(),
  chains = 4,
  iter = 4000,
  seed = 123,
  refresh = 0,
  warmup = floor(iter/2),
  mc.cores = 1,
  control = list(adapt_delta = 0.95),
  return_fit = FALSE,
  quiet = TRUE
)
```

Arguments

`t_event`	Numeric observed time to event t^*_{ei}. Must be `>0`.
`status_event`	Integer vector of event indicators δ_{ei} with values `0` or `1` (`1` = event observed, `0` = right-censored at `t_event`).
`treatment_ind`	Integer vector of treatment assignments x_i with values `0` or `1` (`1` = treated, `0` = control).
`cov`	Optional matrix or data frame of covariates $\boldsymbol{Z}$. Each row corresponds to a subject and each column to a covariate. Covariates must be numeric and are treated as linear effects in the model. Only binary (0/1) and continuous covariates are

supported. Multilevel or categorical variables are not expanded into dummy variables; if provided as numeric (e.g., factor codes), they are treated as continuous. If NULL (default), a no-covariate Weibull model is fitted.

`hyperparams_event`
: A named list of prior hyperparameters for the event-time model. If it is empty (`list()`), default values are assigned internally.
  - `alpha_e`, `beta_e`: Shape and rate parameters of the Gamma prior for the baseline scale parameter λ_{0e} (defaults: 0.1, 0.1).
  - `alpha_re`, `beta_re`: Shape and rate parameters of the Gamma prior for the baseline shape parameter ρ_e (defaults: 0.1, 0.1).
  - `mu_eta`, `sigma_eta`: Mean and standard deviation of the Normal prior for the treatment effect (log hazard ratio) η (defaults: 0, $\sqrt{2}$).
  - `mu_be`, `sigma_be`: Mean and standard deviation of the Normal prior for the covariate effects β_{ej} (defaults: 0, $\sqrt{10}$).

  No other hyperparameters are allowed.

`chains`
: Number of Markov chain Monte Carlo (MCMC) chains. Defaults to `4`.

`iter`
: Number of iterations per chain (including warmup). Defaults to `4000`.

`seed`
: Optional integer seed passed to `sampling` for reproducibility. If `NULL`, 'Stan' generates a seed internally and results may differ across runs. Defaults to `123`.

`refresh`
: Frequency of progress updates from `sampling`. Set to `0` (default) to suppress output.

`warmup`
: Number of warmup (burn-in) iterations per chain. Must be strictly smaller than `iter`. Defaults to `floor(iter / 2)`.

`mc.cores`
: Integer. Number of CPU cores to use when executing Markov chains in parallel via `sampling`. Defaults to `1`. We recommend setting `mc.cores` to the maximum number of processors supported by the available hardware and memory, up to the number of chains.

`control`
: A named list of sampler control parameters passed to `sampling`. Examples include `adapt_delta` and `max_treedepth`. Defaults to `list(adapt_delta = 0.95)`.

`return_fit`
: Logical; if `TRUE`, also return the underlying 'rstan' `stanfit` object. Defaults to `FALSE`.

`quiet`
: Logical. If `TRUE` (default), suppress messages and diagnostic warnings from 'Stan' during model fitting. Useful for large simulation studies.

Details

Let T_{ei} denote the event time for subject i. Let $\boldsymbol{Z}_i$ denote the corresponding covariate vector, which is the i th row of `cov` when provided. Treatment assignment is represented by a known indicator x_i, where for example $x_i = 0$ denotes control and $x_i = 1$ denotes experimental treatment. The treatment effect parameter η represents the log hazard ratio comparing experimental treatment to control, and $\boldsymbol{\beta}_e$ denotes the vector of covariate regression coefficients (log hazard ratios) in the proportional hazards model.

The baseline event time distribution follows a Weibull model with density

$$f_{0e}(t) = \lambda_{0e}\,\rho_e\,t^{\rho_e - 1}\exp(-\lambda_{0e}t^{\rho_e}), \quad t \geq 0.$$

Conditional on treatment and covariates, the hazard function is

$$h_e(t \mid x_i, \boldsymbol{Z}_i) = \lambda_{0e}\, \rho_e\, t^{\rho_e - 1} \exp(\eta x_i + \boldsymbol{Z}_i^\top \boldsymbol{\beta}_e), \quad t \geq 0,$$

and the corresponding survival function is

$$S_e(t \mid x_i, \boldsymbol{Z}_i) = \exp\{-\lambda_{0e}\, t^{\rho_e} \exp(\eta x_i + \boldsymbol{Z}_i^\top \boldsymbol{\beta}_e)\}, \quad t \geq 0.$$

For reference, 'Stan' uses an equivalent Weibull representation based on a shape parameter ρ_e and a scale parameter σ, with baseline density

$$f_{0e}(t) = (\rho_e/\sigma)\,(t/\sigma)^{\rho_e - 1} \exp\{-(t/\sigma)^{\rho_e}\}, \quad t \geq 0.$$

The two formulations are related through

$$\sigma_i = \{\lambda_{0e} \exp(\eta x_i + \boldsymbol{Z}_i^\top \boldsymbol{\beta}_e)\}^{-1/\rho_e}, \quad t \geq 0.$$

Value

A list with the following components:

- `eta`: Posterior draws of the treatment log hazard ratio η.
- `rho_e`: Posterior draws of the Weibull shape parameter ρ_e.
- `lambda_0e`: Posterior draws of the baseline Weibull scale parameter λ_{0e}.
- `beta_e`: Posterior draws of the covariate log hazard ratios $\boldsymbol{\beta}_e$, or `NULL` if no covariates are included.
- `fit`: The 'rstan' `stanfit` object (only if `return_fit = TRUE`).

See Also

Other BayesPET model fitting: `fit_censor`, `fit_enroll`, `fit_event_blind`, `fit_models`, `print.BayesPET_fit`

Examples

```
data(data_example)
example_eventcensor<-data_example$example_eventcensor
# Use chains = 1 here to reduce runtime for the example;
# use more chains in real analyses.
fit_e_unblind <- fit_event_unblind(
  t_event = example_eventcensor$time,
  status_event = example_eventcensor$eventstatus,
  treatment_ind = example_eventcensor$trt,
  cov = example_eventcensor[,6:7],
  chains = 1, iter = 2000, seed = 123,
  return_fit = TRUE
)

summary(fit_e_unblind$eta)
print(fit_e_unblind$fit)
```

`fit_models` *Fit enrollment, event-time, and censoring models to clinical trial data and return posterior draws model parameters*

Description

Fits the enrollment, event-time, and censoring models to trial data and returns posterior draws of model parameters.

Usage

```
fit_models(
  data.enroll,
  data.eventcensor,
  blinded = TRUE,
  p_trt = NULL,
  hyperparams_enroll = list(),
  hyperparams_event = list(),
  hyperparams_censor = list(),
  chains = 4,
  iter = 4000,
  mc.cores = 1,
  warmup = floor(iter/2),
  seed = list(123),
  refresh = 0,
  control = list(list(adapt_delta = 0.95)),
  return_fit = FALSE,
  quiet = TRUE
)
```

Arguments

`data.enroll` A data frame of enrollment information up to the analysis time. Must contain the columns:

- `interarrivaltime`: Numeric vector of interarrival times (> 0).
- `enrollstatus`: Integer vector coded `1` = enrolled, `0` = administratively censored.

Any additional columns are ignored.

`data.eventcensor`

A data frame of observed event/censoring outcomes at the analysis time. Must contain (at minimum) the following columns:

- `time`: observed follow-up time (event or censoring time); must be numeric and > 0.
- `eventstatus`: event indicators (`1` = event, `0` = right-censored).

- `censorstatus`: random censoring indicators δ_{ci} (`1` = random censoring observed, `0` = no random censoring, including administrative censoring or event observed).

If `blinded = FALSE`, `data.eventcensor` must also contain:

- `trt`: observed treatment assignment indicators coded `0/1`.

The column `No` (representing a subject index) may be included but is not required. Any additional columns (other than `No`, `trt`, `time`, `eventstatus`, and `censorstatus`) are treated as numeric baseline covariates and will be used if present.

`blinded` Logical. If `TRUE` (default), the interim analysis is blinded and treatment assignments for current subjects are not observed in the data. If `FALSE`, the analysis is unblinded and observed treatment assignments are used.

`p_trt` Numeric scalar in $[0, 1]$ giving the prespecified randomization probability of assignment to the experimental treatment arm. Required only if `blinded = TRUE`; ignored otherwise. Defaults to `NULL`.

`hyperparams_enroll`

A named list of prior hyperparameters for the enrollment model. If it is empty (`list()`), default values are assigned internally.

- `alpha_mu`: Shape parameter of the Gamma prior for the enrollment rate μ (default: 0.1).
- `beta_mu`: Rate parameter of the Gamma prior for the enrollment rate μ (default: 0.1).

No other hyperparameters are allowed.

`hyperparams_event`

A named list of prior hyperparameters for the event-time model. If it is empty (`list()`), default values are assigned internally.

- `alpha_e`, `beta_e`: Shape and rate parameters of the Gamma prior for the baseline scale parameter λ_{0e} (defaults: 0.1, 0.1).
- `alpha_re`, `beta_re`: Shape and rate parameters of the Gamma prior for the baseline shape parameter ρ_e (defaults: 0.1, 0.1).
- `mu_eta`, `sigma_eta`: Mean and standard deviation of the Normal prior for the treatment effect (log hazard ratio) η (defaults: 0, $\sqrt{2}$).
- `mu_be`, `sigma_be`: Mean and standard deviation of the Normal prior for the covariate effects β_{ej} (defaults: 0, $\sqrt{10}$).

When `blinded = TRUE`, the prior for the treatment effect η is a truncated Normal distribution on $[0, \infty)$ with parameters `mu_eta` and `sigma_eta`. When `blinded = FALSE`, the prior for η is an untruncated Normal distribution. No other hyperparameters are allowed.

`hyperparams_censor`

A named list of prior hyperparameters for the censoring model. If it is empty (`list()`), default values are assigned internally.

- `alpha_c`, `beta_c`: Shape and rate parameters of the Gamma prior for the baseline scale parameter λ_{0c} (defaults: 0.1, 0.1).
- `alpha_rc`, `beta_rc`: Shape and rate parameters of the Gamma prior for the baseline shape parameter ρ_c (defaults: 0.1, 0.1).

- `mu_bc, sigma_bc`: Mean and standard deviation of the Normal prior for the covariate effects β_{cj} (defaults: 0, $\sqrt{10}$).

No other hyperparameters are allowed.

`chains` Number of Markov chain Monte Carlo (MCMC) chains. Defaults to `4`.

`iter` Number of iterations per chain (including warmup). Defaults to `4000`.

`mc.cores` Integer. Number of CPU cores to use when executing Markov chains in parallel via `sampling`. Defaults to `1`. We recommend setting `mc.cores` to the maximum number of processors supported by the available hardware and memory, up to the number of chains.

`warmup` Number of warmup (burn-in) iterations per chain. Must be strictly smaller than `iter`. Defaults to `floor(iter / 2)`.

`seed` Optional random seed(s) passed to `sampling` for reproducibility. Can be specified as:

- a single integer or `NULL`, in which case the same seed is used for all three submodels, or
- a list of up to three integers or `NULL`s, recycled to length 3, corresponding to the enrollment, censoring, and event-time models, respectively.

Use `NULL` to allow 'Stan' to select a seed internally. Defaults to `list(123)`.

`refresh` Frequency of progress updates from `sampling`. Set to `0` (default) to suppress output.

`control` Sampler control settings passed to `sampling` for the three submodels (enrollment, censoring, and event-time). Can be specified as:

- a single named list of control parameters (shared across all three submodels), or
- a list of up to three named lists, recycled to length 3, giving separate control settings for the enrollment, censoring, and event-time models, respectively.

Typical entries include `adapt_delta` and `max_treedepth`.

Defaults to `list(list(adapt_delta = 0.95))`.

`return_fit` Logical; if `TRUE`, also return the underlying 'rstan' `stanfit` objects for the enrollment, censoring, and event models. Defaults to `FALSE`.

`quiet` Logical. If `TRUE` (default), suppress messages and diagnostic warnings from 'Stan' during model fitting. Useful for large simulation studies.

Details

This function fits three submodels: an enrollment model, a censoring model, and an event-time model conditional on the given trial data. If treatment assignments are known, the event-time model is fit using `fit_event_unblind`; otherwise, a blinded event-time model is fit using `fit_event_blind`. Technical details of the likelihoods, priors and parameterizations are documented in `fit_enroll`, `fit_censor`, `fit_event_unblind`, and `fit_event_blind`.

Value

An object of class "BayesPET_fit", a named list containing posterior draws and related information from the fitted models, with elements:

- blinded: Logical; indicates whether the analysis is blinded,
- mu: Posterior draws of the enrollment rate μ.
- rho_c: Posterior draws of the censoring-model Weibull shape parameter ρ_c.
- lambda_0c: Posterior draws of the censoring-model baseline Weibull scale parameter λ_{0c}.
- beta_c: Posterior draws of the censoring-model covariate log hazard ratios $\boldsymbol{\beta}_c$, or NULL if no covariates are included.
- eta: Posterior draws of the treatment log hazard ratio η.
- rho_e: Posterior draws of the event-model Weibull shape parameter ρ_e.
- lambda_0e: Posterior draws of the event-model baseline Weibull scale parameter λ_{0e}.
- beta_e: Posterior draws of the event-model covariate log hazard ratios $\boldsymbol{\beta}_e$, or NULL if no covariates are included.
- treatment_ind: Observed treatment assignments $x_i \in \{0, 1\}$ (only returned if blinded = FALSE).
- x: Posterior draws of latent treatment assignments $x_i \in \{0, 1\}$ (only returned if blinded = TRUE).
- fit: A list with components enroll, censor, and event containing the underlying 'rstan' stanfit objects (present only if return_fit = TRUE).

The default print method displays a concise overview.

See Also

Other BayesPET model fitting: fit_censor, fit_enroll, fit_event_blind, fit_event_unblind, print.BayesPET_fit

Examples

```
data(data_example)
example_enroll <- data_example$example_enroll
example_eventcensor <- data_example$example_eventcensor

# Unblinded analysis
## Use 2 chains and iter = 2000 here to reduce runtime for the example;
## use more chains in real analyses.
fit.unblind <- fit_models(
  data.enroll = example_enroll,
  data.eventcensor = example_eventcensor,
  blinded = FALSE,
  chains = 2, iter = 2000, seed = list(123),
  return_fit = TRUE, mc.cores = 1, quiet = FALSE
)
```

```
# Blinded analysis
example_eventcensor.blind <- example_eventcensor
example_eventcensor.blind$trt <- NA
## Use 2 chains and iter = 2000 here to reduce runtime for the example;
## use more chains in real analyses.
fit.blind <- fit_models(
  data.enroll = example_enroll,
  data.eventcensor = example_eventcensor.blind,
  blinded = TRUE, p_trt = 0.5,
  chains = 2, iter = 2000, seed = list(123),
  return_fit = TRUE, mc.cores = 1, quiet = FALSE
)

print(fit.unblind)
summary(fit.unblind$eta)

print(fit.blind)
summary(fit.blind$eta)
```

`generate_data` *Generate two-arm trial data with enrollment, event, and censoring processes, and return data formatted for event-time prediction.*

Description

Simulates data from a two-arm clinical trial with a time-to-event endpoint. The data generating process incorporates staggered enrollment, event times, and random censoring, with event and censoring distributions specified as Weibull or log-logistic. Treatment and covariate effects are incorporated through a proportional hazards structure.

Usage

```
generate_data(
  N,
  E_target,
  E_cutoff,
  p_trt,
  cov_type,
  cov_dist,
  logHR.trt = NULL,
  enroll_rate,
  dist.event,
  dist.censor,
  blinded = TRUE,
  event.scale = NULL,
  event.shape = NULL,
  censor.scale = NULL,
```

```
  censor.shape = NULL,
  beta.event,
  beta.censor,
  event.scale_trt = NULL,
  event.shape_trt = NULL,
  beta.event_trt = if (is.null(logHR.trt)) beta.event else NULL,
  assess_window = 0,
  seed = 123
)
```

Arguments

N	Integer. Total planned sample size (maximum number of subjects that can be enrolled in the trial).
E_target	Integer. Target number of events for the final analysis.
E_cutoff	Integer. Target number of events for the interim analysis.
p_trt	Scalar randomization probability to the experimental arm, $\gamma \in (0, 1)$.
cov_type	Character vector specifying the distribution for each component of the covariate vector $\boldsymbol{Z} = (Z_1, \ldots, Z_p)$ used in the proportional hazards model. Each element must be "binary" or "continuous". If NULL, no covariates are used.
cov_dist	Numeric vector of the same length as cov_type, giving parameters for the covariate-generating distribution of each Z_j: • "binary": $Z_j \sim \text{Bernoulli}(p_j)$ with $p_j = cov\_dist[j]$. • "continuous": $Z_j \sim N(0, \sigma_j^2)$ with $\sigma_j = cov\_dist[j]$.
logHR.trt	Numeric scalar giving the log hazard ratio for the experimental versus control arm in the event-time model. When NULL (default), the two treatment arms are generated from separate proportional hazards models with arm-specific baseline parameters and covariate effects.
enroll_rate	Positive numeric scalar specifying the enrollment rate.
dist.event	Character. Baseline distribution for event times: "Weibull" or "Loglogistic". This distribution family is the same for both arms.
dist.censor	Character. Baseline distribution for random censoring times: "Weibull" or "Loglogistic".
blinded	Logical. If TRUE (default), the generated interim dataset is blinded and treatment assignments in data.eventcensor$trt are set to NA. If FALSE, treatment assignments in data.eventcensor$trt are coded as 0 for control and 1 for the experimental group.
event.scale	Numeric scalar > 0. Control-arm event baseline scale parameter.
event.shape	Numeric scalar > 0. Control-arm event baseline shape parameter.
censor.scale	Numeric scalar > 0. Random censoring baseline scale parameter.
censor.shape	Numeric scalar > 0. Random censoring baseline shape parameter.
beta.event	Numeric vector. Regression coefficients for baseline covariates in the event-time proportional hazards model; must have the same length and ordering as cov_type.

`beta.censor` Numeric vector. Regression coefficients for baseline covariates in the random censoring-time proportional hazards model; must have the same length and ordering as `cov_type`.

`event.scale_trt` Numeric scalar > 0. Experimental-arm event baseline scale parameter (used when `logHR.trt = NULL`).

`event.shape_trt` Numeric scalar > 0. Experimental-arm event baseline shape parameter (used when `logHR.trt = NULL`).

`beta.event_trt` Numeric vector. Regression coefficients for baseline covariates in the experimental-arm event-time proportional hazards model, used when `logHR.trt = NULL`. Must have the same length and ordering as `cov_type`. Defaults to `beta.event`.

`assess_window` Numeric scalar >= 0. Assessment window width. If > 0, observed event/censoring times are coarsened to the midpoint of the window containing $\min(T_{\text{event}}, T_{\text{censor}})$. Defaults to `0`.

`seed` Integer or `NULL`. Random seed for data generation. If the value is NULL then no random seed is used. Defaults to `123`.

Details

Subjects are randomized independently to the experimental arm with probability `p_trt`. Baseline covariates are generated independently based on `cov_type` and `cov_dist`. Binary covariates follow a Bernoulli distribution, while continuous covariates follow a normal distribution with mean zero and standard deviation determined by `cov_dist`.

Interarrival times between successive enrollments are drawn from an exponential distribution with rate `enroll_rate` with model details documented in `fit_enroll`. Calendar enrollment times are obtained by cumulative summation of these interarrival times.

Event times and random censoring times are generated from Weibull or log-logistic baseline distributions, as specified by `dist.event` and `dist.censor`. For the Weibull model, the baseline survival function is parameterized as

$$S_0(t) = \exp\{-\lambda_0 t^\rho\}, \quad t \geq 0,$$

where $\rho > 0$ is the shape and $\lambda_0 > 0$ is the baseline hazard scale. For the log-logistic model, the baseline survival function is

$$S_0(t) = \{1 + (t/b)^a\}^{-1}, \quad t \geq 0,$$

where $a > 0$ and $b > 0$ denote the shape and scale parameters, respectively. Parameter calibration via marginal median survival can be performed using `convert_median` prior to simulation.

Covariate effects are incorporated through a proportional hazards structure for both the event and censoring processes. When `logHR.trt` is provided, the treatment effect is modeled through a proportional hazards formulation. When `logHR.trt` is `NULL`, the two treatment arms are allowed to differ through separate baseline parameters and covariate effects. The random censoring mechanism does not depend on treatment assignment.

Value

A list with elements:

- `data.enroll`: A data frame of observed enrollment information up to the interim data cut. Columns: subject index `No`, subject enrollment calendar time `enrolltime`, enrollment inter-arrival time `interarrivaltime`, enrollment status `enrollstatus` (1 = enrolled, 0 = administratively censored enrollment process).
- `data.eventcensor`: A data frame of observed survival outcomes at the interim cut. Columns include subject index `No`, treatment assignment indicator `trt` (`NA` if `blinded = TRUE`), observed time `time` (administratively censored at interim cut), event status `eventstatus` (1 = event, 0 = right censored), random censoring status `censorstatus` (1 = random censoring before the interim data cut; 0 = otherwise), followed by covariates.
- `truesurvival`: Full underlying data without administrative censoring: `No`, `trt`, `t_event` (true underlying event time), `t_randcensor` (true underlying random censoring time), `t_event.obs` (underlying follow-up time without administrative censoring), `t_event.obswithintervalassess` (underlying follow-up time without administrative censoring but applied with assessment windows), `status` (1 = event before random censoring), `enrollmenttime`, plus covariates.
- `event.interim.obs`: Observed number of events at the interim data cut. If the prespecified interim event cutoff `E_cutoff` cannot be reached, this equals the maximum number of events observed.
- `event.max`: Number of events that would occur without administrative censoring (i.e., after accounting only for random censoring).
- `cuttime.true`: The true calendar time at which the cumulative number of observed events reaches the target event count. If the target number of events cannot be reached, this is the calendar time of the last observed event or censoring.
- `event.final.obs`: The latent true number of events at `cuttime.true`.

Examples

```
## --- Weibull event/censoring with a common PH treatment effect ---
data.weibull <- generate_data(
  N = 80, E_target = 50, E_cutoff = 25, p_trt = 0.5,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, sqrt(2)),
  beta.event  = c(0.2, 0.2),
  beta.censor = c(0, 0),
  logHR.trt = log(0.5),
  enroll_rate = 50/3, beta.event_trt = NULL,
  dist.event = "Weibull", dist.censor = "Weibull",
  event.scale = 1/5^3, event.shape = 3,
  censor.scale = 10^(-6), censor.shape = 6,
  blinded = TRUE,
  assess_window = 2,
  seed = 1
)
names(data.weibull)
```

```
## --- Log-logistic event/censoring with a common PH treatment effect ---
data.logl <- generate_data(
  N = 80, E_target = 50, E_cutoff = 25, p_trt = 0.5,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, sqrt(2)),
  beta.event  = c(0.2, 0.2),
  beta.censor = c(0, 0),
  logHR.trt = log(0.5),
  enroll_rate = 50/3, beta.event_trt = NULL,
  dist.event = "Loglogistic", dist.censor = "Loglogistic",
  event.scale = 6, event.shape = 6,
  censor.scale = 20, censor.shape = 4,
  blinded = TRUE,
  assess_window = 2,
  seed = 1
)
summary(data.logl$truesurvival$t_event)
### true underlying event time without administrative censoring

## --- Weibull arm-specific models (logHR.trt = NULL) ---
data.weibull.nonPH <- generate_data(
  N = 80, E_target = 50, E_cutoff = 25, p_trt = 0.5,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, sqrt(2)),
  beta.event  = c(0.2, 0.2),
  beta.censor = c(0, 0),
  logHR.trt = NULL,
  enroll_rate = 50/3,
  dist.event = "Weibull", dist.censor = "Weibull",
  event.scale = 1/5^3, event.shape = 3,     # control
  event.scale_trt = 1/6^3, event.shape_trt = 3, # experiment
  beta.event_trt = c(0.15, 0.2),
  censor.scale = 10^(-6), censor.shape = 6,
  blinded = TRUE,
  assess_window = 2,
  seed = 1
)
data.weibull.nonPH$cuttime.true
```

---

get_oc *Generate operating characteristics for event prediction*

Description

Generate operating characteristics for multiple simulated trials

Usage

```
get_oc(
```

```
  N,
  E_target,
  E_cutoff,
  p_trt,
  cov_type,
  cov_dist,
  logHR.trt = NULL,
  enroll_rate,
  dist.event,
  dist.censor,
  blinded = TRUE,
  event.scale = NULL,
  event.shape = NULL,
  censor.scale = NULL,
  censor.shape = NULL,
  beta.event,
  beta.censor,
  event.scale_trt = NULL,
  event.shape_trt = NULL,
  beta.event_trt = if (is.null(logHR.trt)) beta.event else NULL,
  assess_window = 0,
  seed = 123,
  hyperparams_enroll = list(),
  hyperparams_event = list(),
  hyperparams_censor = list(),
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  refresh = 0,
  control = list(list(adapt_delta = 0.95)),
  nsim = 1000,
  nsim.max = 3 * nsim,
  n_workers = 1,
  ...
)
```

Arguments

N	Integer. Total planned sample size (maximum number of subjects that can be enrolled in the trial).
E_target	Integer. Target number of events for the final analysis.
E_cutoff	Integer. Target number of events for the interim analysis.
p_trt	Scalar randomization probability to the experimental arm, $\gamma \in (0, 1)$.
cov_type	Character vector specifying the distribution for each component of the covariate vector $\boldsymbol{Z} = (Z_1, \ldots, Z_p)$ used in the proportional hazards model. Each element must be "binary" or "continuous". If NULL, no covariates are used.
cov_dist	Numeric vector of the same length as cov_type, giving parameters for the covariate-generating distribution of each Z_j:

- "binary": $Z_j \sim \text{Bernoulli}(p_j)$ with $p_j = cov\_dist[j]$.
- "continuous": $Z_j \sim N(0, \sigma_j^2)$ with $\sigma_j = cov\_dist[j]$.

logHR.trt
: Numeric scalar giving the log hazard ratio for the experimental versus control arm in the event-time model. When NULL (default), the two treatment arms are generated from separate proportional hazards models with arm-specific baseline parameters and covariate effects.

enroll_rate
: Positive numeric scalar specifying the enrollment rate.

dist.event
: Character. Baseline distribution for event times: "Weibull" or "Loglogistic". This distribution family is the same for both arms.

dist.censor
: Character. Baseline distribution for random censoring times: "Weibull" or "Loglogistic".

blinded
: Logical. If TRUE (default), the generated interim dataset is blinded and treatment assignments in data.eventcensor$trt are set to NA. If FALSE, treatment assignments in data.eventcensor$trt are coded as 0 for control and 1 for the experimental group.

event.scale
: Numeric scalar > 0. Control-arm event baseline scale parameter.

event.shape
: Numeric scalar > 0. Control-arm event baseline shape parameter.

censor.scale
: Numeric scalar > 0. Random censoring baseline scale parameter.

censor.shape
: Numeric scalar > 0. Random censoring baseline shape parameter.

beta.event
: Numeric vector. Regression coefficients for baseline covariates in the event-time proportional hazards model; must have the same length and ordering as cov_type.

beta.censor
: Numeric vector. Regression coefficients for baseline covariates in the random censoring-time proportional hazards model; must have the same length and ordering as cov_type.

event.scale_trt
: Numeric scalar > 0. Experimental-arm event baseline scale parameter (used when logHR.trt = NULL). Defaults to NULL.

event.shape_trt
: Numeric scalar > 0. Experimental-arm event baseline shape parameter (used when logHR.trt = NULL). Defaults to NULL.

beta.event_trt
: Numeric vector. Regression coefficients for baseline covariates in the experimental-arm event-time proportional hazards model, used when logHR.trt = NULL. Must have the same length and ordering as cov_type. Defaults to beta.event.

assess_window
: Numeric scalar >= 0. Assessment window width. If > 0, observed event/censoring times are coarsened to the midpoint of the window containing $\min(T_{\text{event}}, T_{\text{censor}})$. Defaults to 0.

seed
: Integer. Base random seed used to generate simulated datasets. Replicate i uses seed seed + i - 1. Defaults to 123.

hyperparams_enroll
: List of prior hyperparameters for the enrollment model. See fit_models for details.

hyperparams_event
: List of prior hyperparameters for the event-time model. See fit_models for details.

`hyperparams_censor`
: List of prior hyperparameters for the censoring model. See `fit_models` for details.

`chains`
: Number of Markov chain Monte Carlo (MCMC) chains. Defaults to `4`.

`iter`
: Number of iterations per chain (including warmup). Defaults to `4000`.

`warmup`
: Number of warmup (burn-in) iterations per chain. Must be strictly smaller than `iter`. Defaults to `floor(iter / 2)`.

`refresh`
: Frequency of progress updates from `sampling`. Set to `0` (default) to suppress output.

`control`
: Sampler control settings passed to `sampling` for the three submodels (enrollment, censoring, and event-time). Can be specified as:

  - a single named list of control parameters (shared across all three submodels), or
  - a list of up to three named lists, recycled to length 3, giving separate control settings for the enrollment, censoring, and event-time models, respectively.

  Typical entries include `adapt_delta` and `max_treedepth`.

  Defaults to `list(list(adapt_delta = 0.95))`.

`nsim`
: Integer. Number of valid simulated trial replicates used to compute operating characteristics. Defaults to `1000`.

`nsim.max`
: Integer. Maximum number of simulated datasets to attempt in order to obtain `nsim` valid replicates (i.e., replicates that can reach `E_target`). Defaults to `3*n.sim`.

`n_workers`
: Integer or `NULL`. Number of parallel workers used by the 'future' backend. If `NULL`, defaults to one fewer than the number of available CPU cores.

`...`
: Additional arguments passed to `future_map`. These can be used to control parallel execution behavior (e.g., scheduling, chunk size, or progress reporting) and do not affect data generation or model fitting.

Details

This function first simulates a two-arm time-to-event trial using `generate_data` and constructs interim datasets (`data.enroll` and `data.eventcensor`). It then fits the enrollment, event-time, and random censoring models and generates posterior predictive draws of the calendar time at which the cumulative number of events reaches `E_target` using `predict_eventtime`.

Only simulated datasets in which the target number of events `E_target` is reachable are retained for analysis. Up to `nsim.max` datasets are generated in order to obtain at most `nsim` valid replicates. As a result, the retained replicates correspond to a subset of the attempted simulations, and their associated seeds are recorded explicitly in the returned object.

Value

An object of class `"BayesPET_oc"` containing operating characteristics for event-time prediction, based on simulated trial replicates for which the target number of events `E_target` is reachable. The object is a named list with components:

- `replicate`: A data frame with one row per retained (valid) simulated trial replicate. Columns include:
  - `median`: Median of the posterior predictive draws of the calendar time at which `E_target` events are reached.
  - `cuttime.true`: True calendar time at which the cumulative number of observed events in the simulated trial first reaches `E_target`.
  - `difference`: Absolute prediction error, `abs(median - cuttime.true)`.
- `n_valid`: Number of retained replicates.
- `n_attempt`: Total number of simulated datasets attempted.
- `nsim_target`: Target number of replicates requested.
- `nsim_max`: Maximum number of datasets that may be generated.
- `seed0`: Base random seed used to generate simulated datasets; equals the input argument `seed`.
- `seeds`: Integer vector of seeds corresponding to the retained replicates (one per row of `replicate`).
- `call`: Matched function call.

See Also

Other BayesPET operating characteristics: `summary.BayesPET_oc()`

Examples

```
## Using nsim = 2, chains = 2, and iter = 2000 to reduce runtime.
## Use larger nsim, chains and iter in real analyses.
oc <- get_oc(
  N = 200, E_target = 150,
  E_cutoff = 75, p_trt = 0.5,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, 2),
  beta.event = c(-0.2, -0.2),
  beta.censor = c(0, 0),
  logHR.trt = log(0.65),
  enroll_rate = 16,
  dist.event = "Weibull", dist.censor = "Weibull",
  event.scale = 1/5^3, event.shape = 3,
  censor.scale = 1/10^6, censor.shape = 6,
  blinded = TRUE,
  assess_window = 2,
  seed = 1,
  chains = 2, iter = 2000,
  nsim = 2,
  n_workers = 1
)

summary(oc)
```

plot.BayesPET_predtime

Plot method for BayesPET prediction objects

Description

Plots an object of class "BayesPET_predtime" by displaying a histogram of posterior predictive draws of the calendar time at which the target number of events is reached. Only finite draws are included. A vertical line indicates the posterior median when it is finite.

Usage

```
## S3 method for class 'BayesPET_predtime'
plot(
  x,
  breaks = "Sturges",
  xlab = "Predicted calendar time to reach target number of events",
  ...
)
```

Arguments

x	An object of class "BayesPET_predtime" returned by predict_eventtime.
breaks	Passed to hist. Default is "Sturges".
xlab	X-axis label. Default is "Calendar time to reach E_target".
...	Additional arguments passed to methods. Not used.

Value

Invisibly returns NULL.

See Also

Other BayesPET prediction: predict_eventtime, print.BayesPET_predtime, summary.BayesPET_predtime

Examples

```
data(data_example)
## Reduced number of chains and iterations compared to defaults
## to keep the example computationally manageable.
pred <- predict_eventtime(
  N = 200,
  E_target = 150,
  data.enroll = data_example$example_enroll,
  data.eventcensor = data_example$example_eventcensor,
```

```
  blinded = TRUE,
  p_trt = 0.5,
  chains = 2,
  iter = 2000,
  assess_window = 2,
  seed.fit = 1,
  seed.pred = 2,
  return_fit = TRUE,
  return_draws = TRUE,
  quiet = TRUE
)

print(pred)
summary(pred)
plot(pred)
```

`predict_eventtime` *Predict the calendar time at which a target number of events is reached from interim analysis data*

Description

Fits enrollment, event-time, and random censoring models to data observed at the interim analysis and predicts the calendar time at which the cumulative number of events reaches `E_target`.

Usage

```
predict_eventtime(
  N,
  E_target,
  data.enroll,
  data.eventcensor,
  blinded = TRUE,
  p_trt = NULL,
  hyperparams_enroll = list(),
  hyperparams_event = list(),
  hyperparams_censor = list(),
  chains = 4,
  iter = 4000,
  warmup = floor(iter/2),
  seed.fit = list(123),
  refresh = 0,
  control = list(list(adapt_delta = 0.95)),
  mc.cores = 1,
  assess_window = 0,
  seed.pred = 1,
```

```
  return_fit = FALSE,
  quiet = TRUE,
  return_draws = FALSE
)
```

Arguments

N	Integer. Total planned sample size (maximum number of subjects that can be enrolled in the trial).
E_target	Integer. Target number of events for the final analysis.
data.enroll	A data frame of observed enrollment information up to the interim analysis time. Must contain the columns enrollstatus, enrolltime, and interarrivaltime. These columns follow the conventions defined by generate_data: • enrolltime: Calendar time of enrollment for each subject, measured from the trial time origin. • interarrivaltime: Time between consecutive enrollments. • enrollstatus: Enrollment status indicator with 1 indicating an observed enrollment time and 0 indicating administrative censoring of the enrollment process at the interim analysis time. The No column is optional and provides a subject index used to align enrollment records with data.eventcensor. If missing, it is created internally as No = seq_len(nrow(data.enroll)) and a warning is returned. All subjects in $\mathcal{B}_1$ (subjects enrolled before the interim analysis who have not yet experienced an event or random censoring, as defined from data.eventcensor) must be present in data.enroll and are matched by No. When not all N subjects have enrolled by the interim analysis (i.e., nrow(data.eventcensor) < N), data.enroll must contain exactly one administratively censored enrollment record (enrollstatus == 0). See data_example (element example_enroll) for a concrete example of the expected data layout.
data.eventcensor	A data frame of observed event and censoring outcomes at the interim analysis time. Must contain the columns time, eventstatus, and censorstatus. These columns follow the conventions defined by generate_data: • time: observed follow up time, administratively censored at the interim analysis. • eventstatus: event indicator (1 = event, 0 = right-censored). • censorstatus: random censoring indicators δ_{ci} (1 = random censoring observed, 0 = no random censoring, including administrative censoring or observed event). When blinded = FALSE, the data frame must also contain the treatment assignment indicator column trt coded as 0 or 1. When blinded = TRUE, trt may be present but ignored. The No column is optional and represents a subject index used to align this data frame with data.enroll. If missing, it is created internally as No = 1:nrow(data.eventcensor) and a warning is returned. Any columns other than No, trt, time, eventstatus, and censorstatus are treated as numeric baseline covariates. See fit_models for covariate requirements

`blinded` Logical. If `TRUE` (default), the interim analysis is blinded and treatment assignments for current subjects are not observed in the data. If `FALSE`, the analysis is unblinded and observed treatment assignments are used.

`p_trt` Numeric scalar in $[0, 1]$ giving the prespecified randomization probability of assignment to the experimental treatment arm. Required only if `blinded = TRUE`; ignored otherwise. Defaults to `NULL`.

`hyperparams_enroll` List of prior hyperparameters for the enrollment model. See `fit_models` for details.

`hyperparams_event` List of prior hyperparameters for the event-time model. See `fit_models` for details.

`hyperparams_censor` List of prior hyperparameters for the censoring model. See `fit_models` for details.

`chains` Number of Markov chain Monte Carlo (MCMC) chains. Defaults to `4`.

`iter` Number of iterations per chain (including warmup). Defaults to `4000`.

`warmup` Number of warmup (burn-in) iterations per chain. Must be strictly smaller than `iter`. Defaults to `floor(iter / 2)`.

`seed.fit` Optional random seed(s) passed to `sampling` for reproducibility. Can be specified as:

- a single integer or `NULL`, in which case the same seed is used for all three submodels, or
- a list of up to three integers or `NULL`s, recycled to length 3, corresponding to the enrollment, censoring, and event-time models, respectively.

Use `NULL` to allow 'Stan' to select a seed internally. Defaults to `list(123)`.

`refresh` Frequency of progress updates from `sampling`. Set to `0` (NULL) to suppress output.

`control` Sampler control settings passed to `sampling` for the three submodels (enrollment, censoring, and event-time). Can be specified as:

- a single named list of control parameters (shared across all three submodels), or
- a list of up to three named lists, recycled to length 3, giving separate control settings for the enrollment, censoring, and event-time models, respectively.

Typical entries include `adapt_delta` and `max_treedepth`.

Defaults to `list(list(adapt_delta = 0.95))`.

`mc.cores` Integer. Number of CPU cores to use when executing Markov chains in parallel via `sampling`. Defaults to `1`. We recommend setting `mc.cores` to the maximum number of processors supported by the available hardware and memory, up to the number of chains.

`assess_window` Non-negative numeric. If `> 0`, predicted event/censor times from enrollment are recorded at the midpoint of the assessment window in which they occur. Defaults to `0`.

seed.pred	Optional integer seed for the RNG used in posterior predictive simulation. It controls only the RNG used in posterior predictive simulation and is independent of the 'Stan' sampling seed. Defaults to list(123).
return_fit	Logical; if TRUE, also return the underlying 'rstan' stanfit objects for the enrollment, censoring, and event models. Defaults to FALSE.
quiet	Logical. If TRUE (default), suppress messages and diagnostic warnings from 'Stan' during model fitting. Useful for large simulation studies.
return_draws	Logical. If TRUE, also return the posterior draws from the fitted submodels as result$draws. Defaults to FALSE.

Details

The function fits three components in sequence using fit_models: an enrollment model, an event-time model, and a random censoring model, all based on data observed at the interim analysis. It then performs posterior predictive simulation for two groups of subjects:

- $\mathcal{B}_1$: subjects enrolled before the interim analysis who have not experienced an event or random censoring by the interim data cut;
- $\mathcal{B}_2$: subjects not yet enrolled by the interim analysis.

For each posterior draw, the function estimates the calendar time at which the cumulative number of events reaches E_target.

Value

An object of class "BayesPET_predtime", which is a named list with components:

- prediction: A numeric vector of length S, where S is the number of posterior draws from the fitted models. Each element is a posterior predictive draw of the calendar time at which the cumulative number of events reaches E_target. Values may be Inf if fewer than E_target events occur in that draw.
- fit: Present only if return_fit = TRUE. A list of 'rstan' stanfit objects with components enroll, censor, and event, corresponding to the enrollment, censoring, and event-time models, respectively.
- draws: Present only if return_draws = TRUE. A list of posterior draws of model parameters produced by the fitted submodels. This includes posterior draws from the enrollment, event-time, and censoring models.
- call: The function call used to generate this object.

Methods include print, summary, and plot.

See Also

Other BayesPET prediction: plot.BayesPET_predtime, print.BayesPET_predtime, summary.BayesPET_predtime

Examples

```
data(data_example)
## Reduced number of chains and iterations compared to defaults
## to keep the example computationally manageable.
pred <- predict_eventtime(
  N = 200,
  E_target = 150,
  data.enroll = data_example$example_enroll,
  data.eventcensor = data_example$example_eventcensor,
  blinded = TRUE,
  p_trt = 0.5,
  chains = 2,
  iter = 2000,
  assess_window = 2,
  seed.fit = 1,
  seed.pred = 2,
  return_fit = TRUE,
  return_draws = TRUE,
  quiet = TRUE
)

print(pred)
summary(pred)
plot(pred)
```

---

print.BayesPET_fit *Print method for BayesPET model fitting objects*

Description

Displays a concise overview of an object of class "BayesPET_fit", including whether the analysis is blinded, the number of posterior draws, and the model components.

Usage

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

Arguments

x	An object of class "BayesPET_fit" returned by fit_models.
...	Additional arguments passed to methods. Not used.

Value

The object x, invisibly.

See Also

Other BayesPET model fitting: fit_censor, fit_enroll, fit_event_blind, fit_models, fit_event_unblind

Examples

```
data(data_example)
example_enroll <- data_example$example_enroll
example_eventcensor <- data_example$example_eventcensor

# Blinded analysis
example_eventcensor.blind <- example_eventcensor
example_eventcensor.blind$trt <- NA
## Use 2 chains and iter = 2000 here to reduce runtime for the example;
## use more chains in real analyses.
fit.blind <- fit_models(
  data.enroll = example_enroll,
  data.eventcensor = example_eventcensor.blind,
  blinded = TRUE, p_trt = 0.5,
  chains = 2, iter = 2000, seed = list(123),
  return_fit = TRUE, mc.cores = 1, quiet = FALSE
)
print(fit.blind)
```

---

print.BayesPET_predtime

Print method for BayesPET prediction objects

Description

Displays a brief overview of an object of class "BayesPET_predtime" and lists the components available in the result. For numerical summaries, use summary; for visualization, use plot.

Usage

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

Arguments

x	An object of class "BayesPET_predtime" returned by predict_eventtime.
...	Additional arguments passed to methods. Not used.

Value

The object x, invisibly.

See Also

Other BayesPET prediction: plot.BayesPET_predtime, predict_eventtime, summary.BayesPET_predtime

Examples

```
data(data_example)
## Reduced number of chains and iterations compared to defaults
## to keep the example computationally manageable.
pred <- predict_eventtime(
  N = 200,
  E_target = 150,
  data.enroll = data_example$example_enroll,
  data.eventcensor = data_example$example_eventcensor,
  blinded = TRUE,
  p_trt = 0.5,
  chains = 2,
  iter = 2000,
  assess_window = 2,
  seed.fit = 1,
  seed.pred = 2,
  return_fit = TRUE,
  return_draws = TRUE,
  quiet = TRUE
)

print(pred)
summary(pred)
plot(pred)
```

---

summary.BayesPET_oc *Summary method for BayesPET operating characteristics object*

Description

Computes summary measures of prediction accuracy from a "BayesPET_oc" object. The summaries are based on the differences between the predicted median and true calendar times to reach the target number of events.

Usage

```
## S3 method for class 'BayesPET_oc'
summary(object, thresholds = c(0.25, 0.5, 1, 1.5, 2), ...)

## S3 method for class 'summary.BayesPET_oc'
print(x, digits = 3, ...)
```

Arguments

object	An object of class "BayesPET_oc" returned by get_oc.
thresholds	Numeric vector of non-negative thresholds used to compute the proportion of replicates for which the absolute prediction error is less than each threshold. Defaults to c(0.25, 0.5, 1, 1.5, 2).
...	Not used.
x	An object of class "summary.BayesPET_oc" returned by summary.BayesPET_oc.
digits	Integer specifying the number of decimal places to use when printing numerical summaries. Defaults to 3.

Value

summary.BayesPET_oc returns an object of class "summary.BayesPET_oc", a list containing:

- n_valid: Number of valid simulation replicates (where the target event count can be reached).
- n_attempt: Number of datasets generated to obtain the valid replicates.
- success_rate: Proportion of attempted datasets that produced valid replicates.
- thresholds: Threshold values used to evaluate prediction accuracy.
- pr_lt: Proportion of replicates with prediction error less than each threshold.
- mae: Mean absolute prediction error.
- median_ae: Median absolute prediction error.
- rmse: Root mean squared prediction error.
- call: The matched function call.

print.summary.BayesPET_oc returns the object x, invisibly.

See Also

Other BayesPET operating characteristics: get_oc()

Examples

```
## Using nsim = 2, chains = 2, and iter = 2000 to reduce runtime.
## Use larger nsim, chains and iter in real analyses.
oc <- get_oc(
  N = 200, E_target = 150,
  E_cutoff = 75, p_trt = 0.5,
  cov_type = c("binary", "continuous"),
  cov_dist = c(0.5, 2),
  beta.event = c(-0.2, -0.2),
  beta.censor = c(0, 0),
  logHR.trt = log(0.65),
  enroll_rate = 16,
  dist.event = "Weibull", dist.censor = "Weibull",
  event.scale = 1/5^3, event.shape = 3,
  censor.scale = 1/10^6, censor.shape = 6,
```

```
  blinded = TRUE,
  assess_window = 2,
  seed = 1,
  chains = 2, iter = 2000,
  nsim = 2,
  n_workers = 1
)

summary(oc)
```

---

summary.BayesPET_predtime *Summary method for BayesPET prediction objects*

Description

Summarizes an object of class "BayesPET_predtime" by reporting summary statistics of the posterior predictive distribution of the calendar time at which the target number of events is reached.

Usage

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

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

Arguments

object	An object of class "BayesPET_predtime" returned by predict_eventtime.
...	Additional arguments passed to methods. Not used.
x	An object of class "summary.BayesPET_predtime" returned by summary applied to a "BayesPET_predtime" object.
digits	Integer. Number of significant digits to use when printing numerical summaries. Defaults to 4.

Value

summary.BayesPET_predtime returns an object of class "summary.BayesPET_predtime", which is a named list containing summary information for the posterior predictive distribution of the target event time. Components include:

- S: Total number of posterior predictive draws.
- n_infinite: Number of draws in which the target number of events is not reached.
- prob_not_reached: Proportion of draws in which the target is not reached.

- `q25`: Posterior 25th percent quantile of the target event time.
- `median`: Posterior median of the target event time.
- `q75`: Posterior 75th percent quantile of the target event time.
- `call`: The function call used to generate the prediction.

`print.summary.BayesPET_predtime` returns the object `x`, invisibly.

See Also

Other BayesPET prediction: `plot.BayesPET_predtime`, `predict_eventtime`, `print.BayesPET_predtime`

Examples

```
data(data_example)
## Reduced number of chains and iterations compared to defaults
## to keep the example computationally manageable.
pred <- predict_eventtime(
  N = 200,
  E_target = 150,
  data.enroll = data_example$example_enroll,
  data.eventcensor = data_example$example_eventcensor,
  blinded = TRUE,
  p_trt = 0.5,
  chains = 2,
  iter = 2000,
  assess_window = 2,
  seed.fit = 1,
  seed.pred = 2,
  return_fit = TRUE,
  return_draws = TRUE,
  quiet = TRUE
)

print(pred)
summary(pred)
plot(pred)
```

Index

∗ **BayesPET operating characteristics**
    get_oc, 27
    summary.BayesPET_oc, 39
∗ **datasets**
    data_example, 6

BayesPET *(*BayesPET-package*)*, 2
BayesPET-package, 2

convert_median, 3, *25*

data_example, 6, *34*
dweibull, *5*

fit_censor, 8, *12*, *15*, *18*, *21, 22*, *38*
fit_enroll, *10*, 10, *15*, *18*, *21, 22*, *25*, *38*
fit_event_blind, *10*, *12*, 13, *18*, *21, 22*, *38*
fit_event_unblind, *10*, *12*, *15*, 16, *21, 22*, *38*
fit_models, *10*, *12*, *15*, *18*, 19, *29, 30*, *34–38*
flexsurvreg, *5*
future_map, *30*

generate_data, 23, *30*, *34*
get_oc, 27, *40*

hist, *32*

phreg, *5*
plot, *36*, *38*
plot.BayesPET_predtime, 32, *36*, *39*, *42*
predict_eventtime, *30*, *32*, 33, *38, 39*, *41, 42*
print, *22*, *36*
print.BayesPET_fit, *10*, *12*, *15*, *18*, *22*, 37
print.BayesPET_predtime, *32*, *36*, 38, *42*
print.summary.BayesPET_oc, *40*
print.summary.BayesPET_oc
        *(*summary.BayesPET_oc*)*, 39
print.summary.BayesPET_predtime, *42*
print.summary.BayesPET_predtime
        *(*summary.BayesPET_predtime*)*, 41

rllogis, *5*
rstan, *5*

sampling, *9*, *11, 12*, *14*, *17*, *21*, *30*, *35*
summary, *36*, *38*, *41*
summary.BayesPET_oc, *31*, 39, *40*
summary.BayesPET_predtime, *32*, *36*, *39*, *41*,
        41

uniroot, *4, 5*