Package: BayesTSM (via r-universe)

October 15, 2026

Title Bayesian Progressive Three State Model with Censoring Due to Intervention

Version 1.0.1

Description In screening programs, individuals are usually followed up and tested (screened) for the development of a disease, such as cancer. The target disease often develops progressively in stages; for example healthy (state 1), pre-state disease (state 2), and the disease state (state 3). When the pre-state disease is found during screening it is intervened upon, for example by surgical removal of a lesion, so that the progression of the pre-state disease to disease is interrupted. This is called censoring due to intervention. Researchers often want to estimate the time from baseline to the pre-state disease, the time from the pre-state disease to the disease, and the total time from baseline to the disease. In addition, researchers often want to regress these times on baseline covariates. To these ends, BayesTSM estimates a progressive three-state model with censoring due to intervention using Bayesian estimation methods, as described in Klausch et al. (2023) <doi:10.1214/22-AOAS1669>.

License MIT + file LICENSE

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

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

Encoding UTF-8

Imports actuar, coda, doParallel, foreach, graphics, ggplot2, MASS, MCMCpack, mvtnorm, parallel, posterior, Rcpp, rlang, survival

LinkingTo Rcpp

Depends R (>= 3.5.0)

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Suggests knitr, rmarkdown, bookdown

VignetteBuilder knitr

LazyData true

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

Date/Publication 2026-06-08 18:05:38 UTC

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

RemoteRef HEAD

RemoteSha a829278034f54e35f4ff6b9f6c34d48ba9652e6f

Contents

bayestsm 2
bayestsm_seq 8
gendat 13
get_IC 16
get_IC_seq 17
mod_slice 18
plot.bayestsm 19
plot.ppCIF 19
ppCIF 20
search_prop_sd 24
search_prop_sd_seq 25
summary.bayestsm 26
trim.mcmc 27

Index **28**

`bayestsm` *Bayesian Accelerated Failure-Time Three-State Model with Censoring After Intervention*

Description

Estimates the Bayesian progressive three-state model for interval-censored three-state screening outcomes with censoring after intervention (Klausch et al., 2023). The model is fitted using a Gibbs sampler with data augmentation for the latent event times `X` and `S`, and either slice-sampling or Metropolis-Hastings updates for the transition-model parameters.

Usage

```
bayestsm(
  d = NULL,
  L = NULL,
  R = NULL,
  Z.X = NULL,
  Z.S = Z.X,
  dist.X = "weibull",
  dist.S = "weibull",
```

```
  beta.prior = "t",
  beta.prior.X = 4,
  beta.prior.S = beta.prior.X,
  sig.prior.X = 1,
  sig.prior.S = 1,
  fix.sigma.X = FALSE,
  fix.sigma.S = FALSE,
  mc = 10000,
  chains = 2,
  thinning = 1,
  warmup = mc * 0.05,
  prop.sd = NULL,
  update_till_convergence = FALSE,
  mc_update = mc,
  min_R = 1.01,
  min_eff = chains * 100,
  maxit = NULL,
  prev.run = NULL,
  MH = FALSE,
  rescale_times = TRUE,
  slicesampler_stepsize = 1,
  silent = FALSE,
  log_prior_fun = log_aft_prior,
  do_cpp = TRUE,
  seed_chains = NULL
)
```

Arguments

`d`	Numeric vector indicating the observed outcome category for each individual. Allowed values are `1`, `2`, and `3`.
`L`	Numeric vector of left interval bounds. Minimum allowed value is `0`.
`R`	Numeric vector of right interval bounds. Use `Inf` for right-censored observations. If `R = Inf`, `d` must be `1`. If `R < Inf`, `d` must be `2` or `3`.
`Z.X`	Optional design matrix of covariates for the first transition model for latent time `X`.
`Z.S`	Optional design matrix of covariates for the second transition model for latent time `S`. Defaults to `Z.X`.
`dist.X`	Character string specifying the distribution for the `X` transition model. Allowed values are `"weibull"`, `"lognormal"`, and `"loglog"`.
`dist.S`	Character string specifying the distribution for the `S` transition model. Allowed values are `"weibull"`, `"lognormal"`, and `"loglog"`.
`beta.prior`	Character string specifying the default prior family for regression coefficients when `log_prior_fun = log_aft_prior`. Allowed values are `"t"` for a Student-t prior and `"norm"` for a normal prior. Default is `"t"`.
`beta.prior.X`	Prior hyperparameter for the regression coefficients in the `X` transition model. If `log_prior_fun = log_aft_prior` and `beta.prior = "t"`, this is the degrees

of freedom of the Student-t prior. If `beta.prior = "norm"`, this is the standard deviation of the normal prior. For custom prior functions, this value is passed as `tau`. Default is `4`.

`beta.prior.S` Prior hyperparameter for the regression coefficients in the `S` transition model. Interpreted in the same way as `beta.prior.X`. For custom prior functions, this value is passed as `tau`. Defaults to `beta.prior.X`.

`sig.prior.X` Prior hyperparameter for the scale parameter `sigma_X` of the `X` transition model. When `fix.sigma.X = FALSE`, this value is passed as `sig.prior` to the prior function for the `X` model. When `fix.sigma.X = TRUE`, it is interpreted as the fixed value of `sigma_X`. Default is `1`.

`sig.prior.S` Prior hyperparameter for the scale parameter `sigma_S` of the `S` transition model. Interpreted in the same way as `sig.prior.X`. When `fix.sigma.S = TRUE`, this is the fixed value of `sigma_S`. Default is `1`.

`fix.sigma.X` Logical; whether to fix the scale parameter `sigma_X` at `sig.prior.X`.

`fix.sigma.S` Logical; whether to fix the scale parameter `sigma_S` at `sig.prior.S`.

`mc` Integer; number of Gibbs iterations per chain.

`chains` Integer; number of MCMC chains.

`thinning` Integer; thinning interval applied when constructing the post-sampling output.

`warmup` Numeric; number of original Gibbs iterations to omit as warmup before assessing convergence. Internally this is converted to stored post-thinning iterations using `round(warmup / thinning)`. `warmup` should be significantly smaller than `mc`; default `mc * 0.05`.

`prop.sd` Proposal scale for the random-walk Metropolis-Hastings update. Used only when `MH = TRUE`. This may be a proposal standard deviation or proposal covariance specification used for the parameter updates. If `NULL` and `MH = TRUE`, the function runs a short preliminary sampler and searches for a useful proposal scale automatically.

`update_till_convergence` Logical; whether to automatically extend the MCMC run until the convergence criteria based on `min_R` and `min_eff` are met, or until `maxit` is reached.

`mc_update` Integer; number of Gibbs iterations per chain to add when updating a previous run or when automatic convergence updating is enabled. Defaults to `mc`.

`min_R` Numeric; target upper threshold for convergence based on the potential scale reduction factor. Used when `update_till_convergence = TRUE`. Default is `1.1`.

`min_eff` Numeric; target minimum effective sample size. Used when `update_till_convergence = TRUE`. Default is `chains * 100`.

`maxit` Optional integer; maximum number of automatic update cycles when `update_till_convergence = TRUE`. If `NULL`, no explicit maximum number of update cycles is imposed by this argument.

`prev.run` Optional fitted object returned by a previous call to `bayestsm()`. If supplied, the sampler continues from the last sampled parameter values and reuses the stored data, priors, proposal settings, distributions, fixed-scale settings, and other model options from the previous run.

MH	Logical; whether to use random-walk Metropolis-Hastings updates for the transition-model parameters. If FALSE, slice-sampling updates are used. Default is FALSE.
rescale_times	Logical; whether to rescale L and R by the median of the finite values of R before sampling. Returned data are stored on the original time scale. Default is TRUE.
slicesampler_stepsize	Numeric step-out size used by the slice sampler when MH = FALSE. Default is 1.
silent	Logical; whether to suppress progress messages.
log_prior_fun	Function used to evaluate the log-prior density for the AFT transition-model parameters. The default is log_aft_prior. Custom prior functions should accept the arguments eta, tau, and sig.prior; see log_aft_prior() for details.
do_cpp	Logical; whether to use the C++ implementation of the Gibbs sampler. If FALSE, the R implementation is used. The samplers are equivalent, but the C++ implementation is faster.
seed_chains	Integer; A vector of length length(chains) with seeds to which the MCMC chains will be initialized using set.seed. If NULL, random seeds are generated.

Details

The function fits a Bayesian accelerated failure-time model for progressive three-state screening outcomes with censoring after intervention. The time from state 1 to state 2 is denoted X, and the time from state 2 to state 3 is denoted S. In each Gibbs iteration, the latent transition times are updated by data augmentation conditional on the current parameter values, and the transition-model parameters are updated using either slice-sampling or random-walk Metropolis-Hastings steps.

The observed data are supplied through the event indicator d and interval bounds L and R, with optional covariate matrices Z.X and Z.S. The distributions of the latent times X and S are specified by dist.X and dist.S.

By default, L and R are divided by the median of the finite right interval bounds before sampling. The fitted object stores the observed data on the original time scale. Klausch et al. (2023) did not rescale time. However, in an AFT model the intercept is measured on the log-time scale, so a fixed default prior on the intercept can have different shrinkage behavior under arbitrary choices of the time unit. Rescaling by a typical observed time scale makes the default prior less sensitive to the units in which time is supplied. This behavior can be disabled with rescale_times = FALSE.

The default prior is evaluated by log_aft_prior(). For each transition model, the AFT parameter vector contains the intercept first, followed by the slope coefficients, with log(sigma) in the final position. With log_prior_fun = log_aft_prior and beta.prior = "t", independent Student-t priors are assigned to the regression coefficients. The degrees of freedom are given by beta.prior.X for the X model and beta.prior.S for the S model. With beta.prior = "norm", independent normal priors are assigned to the regression coefficients, and beta.prior.X and beta.prior.S are interpreted as prior standard deviations.

The scale parameters sigma_X and sigma_S can either be estimated or fixed. When fix.sigma.X = FALSE and fix.sigma.S = FALSE, the respective scale parameters are estimated. Under the default prior function, normal prior kernels restricted to positive values are used for sigma_X and sigma_S, with prior standard deviations sig.prior.X and sig.prior.S, respectively. Equivalently, these are half-normal priors up to an additive constant in the log-prior. The Jacobian adjustment for the log(sigma) parameterization is included by the prior function. When fix.sigma.X = TRUE or fix.sigma.S = TRUE, the corresponding scale parameter is fixed throughout sampling,

with sig.prior.X and/or sig.prior.S giving the fixed value. For example, fix.sigma.S = TRUE with sig.prior.S = 1 fixes sigma_S at 1 during estimation. If, in addition, dist.S = "weibull" is chosen, the latent S transition follows the exponential special case of the Weibull model.

Custom prior functions can be supplied through log_prior_fun. The same prior function is currently used for the X and S transition models, but the prior parameters may differ between the two models. Specifically, beta.prior.X and sig.prior.X are passed as tau and sig.prior to the prior function for the X model, while beta.prior.S and sig.prior.S are passed as tau and sig.prior to the prior function for the S model. Additional prior parameters, if needed, should currently be hardcoded inside the custom prior function. See log_aft_prior() for details on the required parameter vector and on how to modify the default prior.

Klausch et al. (2023) used random-walk Metropolis-Hastings updates for the transition-model parameters. This can be requested with MH = TRUE. The current default is MH = FALSE, which uses a slice sampler instead. In practice, the slice sampler has shown faster convergence. The default step-out size slicesampler_stepsize = 1 usually works well, but it can be increased or decreased if the slice sampler needs further tuning.

Passing a previous fitted object through prev.run continues an earlier MCMC run rather than starting from scratch. In that case, the function initializes from the last sampled parameter values of the previous run and reuses the stored data, covariates, priors, proposal settings, distributions, fixed-scale settings, prior function, sampler choice, and controller settings.

If update_till_convergence = TRUE, the function checks convergence after the initial run and repeatedly extends the sampler by mc_update iterations per chain until the convergence criteria based on min_R and min_eff are satisfied, or until maxit is reached. Progress is printed after each extension run of mc_update iterations per chain.

Value

A list with the following components:

par.X An mcmc.list containing the stored sampled draws of the X transition-model parameters for all chains. The final column is returned on the natural scale as sigma.

par.S An mcmc.list containing the stored sampled draws of the S transition-model parameters for all chains. The final column is returned on the natural scale as sigma.

X Numeric vector containing sampled latent X values, concatenated over chains.

S Numeric vector containing sampled latent S values, concatenated over chains.

ac Acceptance indicators for the Metropolis-Hastings parameter updates.

ac.cur Acceptance indicators from the current extension run only, returned when prev.run is supplied.

dat A data.frame containing the data used in fitting, with columns d, L, and R on the original time scale.

priors A list containing the prior hyperparameters beta.prior.X, beta.prior.S, sig.prior.X, and sig.prior.S.

thinning The thinning interval used to construct par.X and par.S. For long MCMC chains, larger thinning values reduce memory burden.

Z.S The covariate matrix used for the S transition model.

Z.X The covariate matrix used for the X transition model.

`prop.sd` The proposal tuning specification used for the Metropolis-Hastings updates.

`dist.X` The distribution used for the `X` transition model.

`dist.S` The distribution used for the `S` transition model.

`fix.sigma.X` Logical; whether `sigma_X` was fixed during fitting.

`fix.sigma.S` Logical; whether `sigma_S` was fixed during fitting.

`warmup` The number of original Gibbs iterations omitted as warmup before convergence assessment.

`beta.prior` The prior family used by the default prior function, either `"t"` or `"norm"`.

`seed_chains` Integer vector of random seeds used for the chains.

`runtime` Total runtime accumulated over the initial run and any extension runs.

`update_till_convergence` Logical; whether automatic convergence updating was requested.

`min_R` The convergence threshold used for the potential scale reduction factor.

`min_eff` The minimum effective sample size threshold used for convergence assessment.

`silent` Logical; whether progress messages were suppressed.

`do_cpp` Logical; whether the C++ sampler was used.

`MH` Logical; whether Metropolis-Hastings updates were used.

`slicesampler_stepsize` Step-out size used by the slice sampler.

`log_prior_fun` Prior function used for the AFT transition-model parameters.

`rescale_times` Logical; whether to rescale input times by the median of finite `R`.

`n_update` Integer; counter of the number of times the model was updated.

`seed_chains` Integer; seeds set at initialization of the Gibbs sampler.

References

Klausch, T., Akwiwu, E. M. U., van de Wiel, M. A., Coupé, V. M. H., and Berkhof, J. (2023). A Bayesian accelerated failure time model for interval censored three-state screening outcomes. *The Annals of Applied Statistics*, 17(2), 1285–1306. doi:10.1214/22AOAS1669

See Also

`log_aft_prior()` `plot.bayestsm()` `ppCIF()` `search_prop_sd()` `summary.bayestsm()` `trim.mcmc()`

Examples

```
library(BayesTSM)

# Generate data
dat = gendat(
             n = 1000,  # Sample size
             p = 2,     # Number of normal distributed covariates
             r = 0,     # Correlation of the covariates
             sigma.X = 0.3,         # True scale parameter
             mu.X    = 2,           # True intercept parameter
             beta.X  = c(0.5,0.5),  # True slope parameters
```

```
              sigma.S = 0.5,         # True scale parameter
              mu.S    = 1,           # True intercept parameters
              beta.S  = c(0.5,0.5),  # True slope parameters
              dist.X  = 'weibull',   # Distribution of X
              dist.S  = 'weibull',   # Distribution of S
              v.min   = 1,           # Min time between screening moments
              v.max   = 5,           # Max time between screening moments
              Tmax    = 2e2,         # Max number of screening times
              mean.rc = 10           # Mean time to right censoring
             )

# Run bayestsm Gibbs sampler with data augmentation and slice sampling of the parameters
# Toy run with few posterior draws
mod_slice   = bayestsm(
              d              = dat$d,
              L              = dat$L,
              R              = dat$R,
              Z.X            = dat[,c('Z.1','Z.2')],
              Z.S            = dat[,c('Z.1','Z.2')],
              mc             = 1e2,             # Increase
              warmup         = 10,              # Discarded before assessing convergence
              thinning       = 1,               # The chain can be thinned to save memory
              chains         = 2,                # In practice use more than 2 chains
              update_till_convergence = FALSE,
              MH             = FALSE,            # Set to TRUE for Metropolis sampling
              dist.X         = 'weibull',        # Correctly specified distributions
              dist.S         = 'weibull'
              )

# Run bayestsm Gibbs sampler with data augmentation and Metropolis sampling of the parameters

mod_MH      = bayestsm(
              d              = dat$d,
              L              = dat$L,
              R              = dat$R,
              Z.X            = dat[,c('Z.1','Z.2')],
              Z.S            = dat[,c('Z.1','Z.2')],
              mc             = 1e2,             # Increase
              warmup         = 10,              # discarded before assessing convergence
              thinning       = 1,               # The chain can be thinned to save memory
              chains         = 2,                # In practice use more than 2 chains
              prop.sd        = 0.01,             # If NULL searched in pilot run
              update_till_convergence = FALSE,
              MH             = TRUE,             # set to TRUE for Metropolis sampling
              dist.X         = 'weibull',        # Correctly specified distributions
              dist.S         = 'weibull'
              )
```

`bayestsm_seq` *Bayesian Accelerated Failure-Time Three-State Model with Censoring After Intervention (sequential processing)*

Description

Estimates the Bayesian progressive three-state model for interval-censored three-state screening outcomes with censoring after intervention (Klausch et al., 2023). The model is fitted using a Gibbs sampler with data augmentation for the latent event times `X` and `S`, and either slice-sampling or Metropolis-Hastings updates for the transition-model parameters. MCMC chains are processed sequentially.

Usage

```
bayestsm_seq(
  d = NULL,
  L = NULL,
  R = NULL,
  Z.X = NULL,
  Z.S = Z.X,
  dist.X = "weibull",
  dist.S = "weibull",
  beta.prior = "t",
  beta.prior.X = 4,
  beta.prior.S = beta.prior.X,
  sig.prior.X = 1,
  sig.prior.S = 1,
  fix.sigma.X = FALSE,
  fix.sigma.S = FALSE,
  mc = 10000,
  chains = 2,
  thinning = 1,
  warmup = mc * 0.05,
  prop.sd = NULL,
  update_till_convergence = FALSE,
  mc_update = mc,
  min_R = 1.01,
  min_eff = chains * 100,
  maxit = NULL,
  prev.run = NULL,
  MH = FALSE,
  rescale_times = TRUE,
  slicesampler_stepsize = 1,
  silent = FALSE,
  log_prior_fun = log_aft_prior,
  do_cpp = TRUE,
  seed_chains = NULL
)
```

Arguments

d	Numeric vector indicating the observed outcome category for each individual. Allowed values are 1, 2, and 3.
L	Numeric vector of left interval bounds. Minimum allowed value is 0.
R	Numeric vector of right interval bounds. Use Inf for right-censored observations. If R = Inf, d must be 1. If R < Inf, d must be 2 or 3.
Z.X	Optional design matrix of covariates for the first transition model for latent time X.
Z.S	Optional design matrix of covariates for the second transition model for latent time S. Defaults to Z.X.
dist.X	Character string specifying the distribution for the X transition model. Allowed values are "weibull", "lognormal", and "loglog".
dist.S	Character string specifying the distribution for the S transition model. Allowed values are "weibull", "lognormal", and "loglog".
beta.prior	Character string specifying the default prior family for regression coefficients when log_prior_fun = log_aft_prior. Allowed values are "t" for a Student-t prior and "norm" for a normal prior. Default is "t".
beta.prior.X	Prior hyperparameter for the regression coefficients in the X transition model. If log_prior_fun = log_aft_prior and beta.prior = "t", this is the degrees of freedom of the Student-t prior. If beta.prior = "norm", this is the standard deviation of the normal prior. For custom prior functions, this value is passed as tau. Default is 4.
beta.prior.S	Prior hyperparameter for the regression coefficients in the S transition model. Interpreted in the same way as beta.prior.X. For custom prior functions, this value is passed as tau. Defaults to beta.prior.X.
sig.prior.X	Prior hyperparameter for the scale parameter sigma_X of the X transition model. When fix.sigma.X = FALSE, this value is passed as sig.prior to the prior function for the X model. When fix.sigma.X = TRUE, it is interpreted as the fixed value of sigma_X. Default is 1.
sig.prior.S	Prior hyperparameter for the scale parameter sigma_S of the S transition model. Interpreted in the same way as sig.prior.X. When fix.sigma.S = TRUE, this is the fixed value of sigma_S. Default is 1.
fix.sigma.X	Logical; whether to fix the scale parameter sigma_X at sig.prior.X.
fix.sigma.S	Logical; whether to fix the scale parameter sigma_S at sig.prior.S.
mc	Integer; number of Gibbs iterations per chain.
chains	Integer; number of MCMC chains.
thinning	Integer; thinning interval applied when constructing the post-sampling output.
warmup	Numeric; number of original Gibbs iterations to omit as warmup before assessing convergence. Internally this is converted to stored post-thinning iterations using round(warmup / thinning). warmup should be significantly smaller than mc; default mc * 0.05.

`prop.sd` Proposal scale for the random-walk Metropolis-Hastings update. Used only when `MH = TRUE`. This may be a proposal standard deviation or proposal covariance specification used for the parameter updates. If `NULL` and `MH = TRUE`, the function runs a short preliminary sampler and searches for a useful proposal scale automatically.

`update_till_convergence` Logical; whether to automatically extend the MCMC run until the convergence criteria based on `min_R` and `min_eff` are met, or until `maxit` is reached.

`mc_update` Integer; number of Gibbs iterations per chain to add when updating a previous run or when automatic convergence updating is enabled. Defaults to `mc`.

`min_R` Numeric; target upper threshold for convergence based on the potential scale reduction factor. Used when `update_till_convergence = TRUE`. Default is `1.1`.

`min_eff` Numeric; target minimum effective sample size. Used when `update_till_convergence = TRUE`. Default is `chains * 100`.

`maxit` Optional integer; maximum number of automatic update cycles when `update_till_convergence = TRUE`. If `NULL`, no explicit maximum number of update cycles is imposed by this argument.

`prev.run` Optional fitted object returned by a previous call to `bayestsm()`. If supplied, the sampler continues from the last sampled parameter values and reuses the stored data, priors, proposal settings, distributions, fixed-scale settings, and other model options from the previous run.

`MH` Logical; whether to use random-walk Metropolis-Hastings updates for the transition-model parameters. If `FALSE`, slice-sampling updates are used. Default is `FALSE`.

`rescale_times` Logical; whether to rescale `L` and `R` by the median of the finite values of `R` before sampling. Returned data are stored on the original time scale. Default is `TRUE`.

`slicesampler_stepsize` Numeric step-out size used by the slice sampler when `MH = FALSE`. Default is `1`.

`silent` Logical; whether to suppress progress messages.

`log_prior_fun` Function used to evaluate the log-prior density for the AFT transition-model parameters. The default is `log_aft_prior`. Custom prior functions should accept the arguments `eta`, `tau`, and `sig.prior`; see `log_aft_prior()` for details.

`do_cpp` Logical; whether to use the C++ implementation of the Gibbs sampler. If `FALSE`, the R implementation is used. The samplers are equivalent, but the C++ implementation is faster.

`seed_chains` Integer; A vector of length `length(chains)` with seeds to which the MCMC chains will be initialized using `set.seed`. If `NULL`, random seeds are generated.

Details

See `bayestsm()`. Contrary to `bayestsm()`, the MCMC chains are run sequentially using a simple loop over chains. No parallel backend is started or required. All else is the same.

Value

A list with the following components:

`par.X` An `mcmc.list` containing the stored sampled draws of the `X` transition-model parameters for all chains. The final column is returned on the natural scale as `sigma`.

`par.S` An `mcmc.list` containing the stored sampled draws of the `S` transition-model parameters for all chains. The final column is returned on the natural scale as `sigma`.

`X` Numeric vector containing sampled latent `X` values, concatenated over chains.

`S` Numeric vector containing sampled latent `S` values, concatenated over chains.

`ac` Acceptance indicators for the Metropolis-Hastings parameter updates.

`ac.cur` Acceptance indicators from the current extension run only, returned when `prev.run` is supplied.

`dat` A `data.frame` containing the data used in fitting, with columns `d`, `L`, and `R` on the original time scale.

`priors` A list containing the prior hyperparameters `beta.prior.X`, `beta.prior.S`, `sig.prior.X`, and `sig.prior.S`.

`thinning` The thinning interval used to construct `par.X` and `par.S`. For long MCMC chains, larger `thinning` values reduce memory burden.

`Z.S` The covariate matrix used for the `S` transition model.

`Z.X` The covariate matrix used for the `X` transition model.

`prop.sd` The proposal tuning specification used for the Metropolis-Hastings updates.

`dist.X` The distribution used for the `X` transition model.

`dist.S` The distribution used for the `S` transition model.

`fix.sigma.X` Logical; whether `sigma_X` was fixed during fitting.

`fix.sigma.S` Logical; whether `sigma_S` was fixed during fitting.

`warmup` The number of original Gibbs iterations omitted as warmup before convergence assessment.

`beta.prior` The prior family used by the default prior function, either `"t"` or `"norm"`.

`seed_chains` Integer vector of random seeds used for the chains.

`runtime` Total runtime accumulated over the initial run and any extension runs.

`update_till_convergence` Logical; whether automatic convergence updating was requested.

`min_R` The convergence threshold used for the potential scale reduction factor.

`min_eff` The minimum effective sample size threshold used for convergence assessment.

`silent` Logical; whether progress messages were suppressed.

`do_cpp` Logical; whether the C++ sampler was used.

`MH` Logical; whether Metropolis-Hastings updates were used.

`slicesampler_stepsize` Step-out size used by the slice sampler.

`log_prior_fun` Prior function used for the AFT transition-model parameters.

`rescale_times` Logical; whether to rescale input times by the median of finite `R`.

`n_update` Integer; counter of the number of times the model was updated.

`seed_chains` Integer; seeds set at initialization of the Gibbs sampler.

References

Klausch, T., Akwiwu, E. M. U., van de Wiel, M. A., Coupé, V. M. H., and Berkhof, J. (2023). A Bayesian accelerated failure time model for interval censored three-state screening outcomes. *The Annals of Applied Statistics*, 17(2), 1285–1306. doi:10.1214/22AOAS1669

See Also

`log_aft_prior()` `plot.bayestsm()` `ppCIF()` `search_prop_sd()` `summary.bayestsm()` `trim.mcmc()`

`gendat` *Generate interval-censored three-state screening data*

Description

Simulates interval-censored three-state screening data with covariates, latent transition times, screening visit schedules, and observed event categories as described by Klausch et al. (2023).

Usage

```
gendat(
  n,
  p = 3,
  p.discrete = 0,
  r = 0.1,
  s = 1,
  sigma.X = 1/2,
  mu.X = 1,
  beta.X = NULL,
  sigma.S = 1/2,
  mu.S = 1,
  beta.S = NULL,
  cor.X.S = 0,
  Tmax = 20,
  v.min = 1,
  v.max = 6,
  mean.rc = 40,
  dist.X = "weibull",
  dist.S = "weibull",
  do.b.XS = FALSE,
  b.XS = NULL
)
```

Arguments

`n`	Integer; number of individuals to simulate.
`p`	Integer; number of multivariate normally distributed covariates.

p.discrete	Integer; number of additional discrete covariates. Currently only 0 or 1 is supported.
r	Common correlation parameter used for the continuous covariates.
s	Standard deviation used for each continuous covariate.
sigma.X	Scale parameter for the first transition model (X).
mu.X	Intercept for the first transition model (X).
beta.X	Numeric vector of regression coefficients for the continuous and discrete covariates in the first transition model. Should have length equal to the total number of covariates.
sigma.S	Scale parameter for the second transition model (S).
mu.S	Intercept for the second transition model (S).
beta.S	Numeric vector of regression coefficients for the continuous and discrete covariates in the second transition model. Should have length equal to the total number of covariates.
cor.X.S	Correlation between the latent errors of X and S when dist.X = "bv-lognormal".
Tmax	Integer; maximum number of screening visits generated per individual.
v.min	Minimum gap between two consecutive screening visits.
v.max	Maximum gap between two consecutive screening visits.
mean.rc	Mean of the exponential random censoring time distribution.
dist.X	Character string specifying the distribution used to simulate X. Supported values are "weibull", "loglog", "lognormal", and "bv-lognormal". If dist.X = "bv-lognormal", then X and S are simulated jointly from a bivariate log-normal model with latent-error correlation cor.X.S, and the value of dist.S is ignored.
dist.S	Character string specifying the distribution used to simulate S. Supported values are "weibull", "loglog", and "lognormal". This argument is ignored when dist.X = "bv-lognormal".
do.b.XS	Logical; whether to include a dependence term proportional to log(X) in the linear predictor for S.
b.XS	Numeric coefficient multiplying log(X) when do.b.XS = TRUE.

Details

The function first simulates covariates from a multivariate normal distribution with exchangeable correlation structure and optional additional Bernoulli covariate. Given these covariates, latent transition times X and S are generated from accelerated failure time models on the log-time scale.

The total time to the second event is X + S. Individuals are then assigned a sequence of screening visit times. The first screening time is sampled uniformly on the interval from v.min to v.max, and each subsequent screening time is generated by adding a uniform increment between v.min and v.max to the previous visit time. Follow-up is administratively censored at a random censoring time generated from an exponential distribution with mean mean.rc.

The observed data are constructed from the screening interval in which the first detected transition occurs. The event indicator d equals 2 if the first transition time X falls in the observed interval but the second event time X + S does not; d equals 3 if both transitions have occurred by the same

observed screening interval; and d equals 1 if no event is observed during follow-up, in which case R = Inf.

When do.b.XS = TRUE, the second transition time depends additionally on log(X) through coefficient b.XS. This induces dependence between the two transition times beyond the shared covariate effects. Alternatively, when dist.X = "bv-lognormal", both X and S are generated jointly from a bivariate log-normal construction with correlation cor.X.S; in that case the separate specification through dist.S is overridden.

This function is primarily intended for simulation studies, examples, and testing of the model-fitting functions in the package.

Value

A data.frame with one row per individual. It contains:

- L Left endpoint of the observed event interval.
- R Right endpoint of the observed event interval. Right-censored observations have R = Inf.
- d Observed event category.
- X Simulated latent time from baseline to the first transition.
- S Simulated latent time from the first to the second transition.
- Z Simulated covariates, included when p > 0.

Examples

```
dat = gendat(
              n = 1000,  # Sample size
              p = 2,     # Number of normal distributed covariates
              r = 0,     # Correlation of the covariates
              sigma.X = 0.3,         # True scale parameter
              mu.X    = 2,           # True intercept parameter
              beta.X  = c(0.5,0.5),  # True slope parameters
              sigma.S = 0.5,         # True scale parameter
              mu.S    = 1,           # True intercept parameters
              beta.S  = c(0.5,0.5),  # True slope parameters
              dist.X  = 'weibull',   # Distribution of X
              dist.S  = 'weibull',   # Distribution of S
              v.min   = 1,           # Min time between screening moments
              v.max   = 5,           # Max time between screening moments
              Tmax    = 2e2,         # Max number of screening times
              mean.rc = 10           # Mean time to right censoring
            )
```

get_IC *Compute information criteria for a fitted* bayestsm *model*

Description

Computes DIC, WAIC1, and WAIC2 for a fitted bayestsm model from posterior draws of the model parameters.

Usage

```
get_IC(mod, samples = nrow(mod$par.X[[1]]) - 1, warmup = NULL, cores = 2)
```

Arguments

mod	A fitted model object returned by bayestsm().
samples	Integer; number of posterior draws used to approximate the information criteria. Defaults to the number of post-burn-in draws in the first chain of mod$par.X after removing the first warmup iterations per chain. Has to be smaller than nrow(mod$par.X[[1]]) minus warmup.
warmup	The number of iterations discarded for warmup per chain. The default is NULL in which case, conservatively, the first half of posterior samples is discarded.
cores	Optional integer; number of cores used for parallel evaluation of the observation-level likelihood contributions. Defaults to 2. To set to maximum available cores set to NULL.

Details

The function by removes the first warmup iteration from the MCMC output stored in mod$par.X and mod$par.S, then samples samples posterior draws from the remaining iterations. If warmup is not specified, the first half of draws is removed, which is a conservative choice. For each sampled draw, the observation-level likelihood contributions are computed, and these are combined into the reported information criteria.

Lower values of DIC, WAIC1, and WAIC2 indicate better expected predictive fit, but these criteria should primarily be compared between models fitted to the same data.

Value

A numeric matrix with one row and three columns:

WAIC1 The first WAIC estimate based on the mean log pointwise posterior density and a bias correction using the average log likelihood.

WAIC2 The second WAIC estimate based on the mean log pointwise posterior density and a bias correction using the posterior variance of the log likelihood contributions.

DIC The deviance information criterion.

See Also

bayestsm()

Examples

```
data(mod_slice) # A toy example model with only 100 posterior draws

get_IC(mod_slice, warmup = 10)
```

---

get_IC_seq *Compute information criteria for a fitted* bayestsm *model (sequential processing)*

Description

Computes DIC, WAIC1, and WAIC2 for a fitted bayestsm model from posterior draws of the model parameters.

Usage

```
get_IC_seq(mod, samples = nrow(mod$par.X[[1]]) - 1, warmup = NULL, cores = 2)
```

Arguments

mod	A fitted model object returned by bayestsm().
samples	Integer; number of posterior draws used to approximate the information criteria. Defaults to the number of post-burn-in draws in the first chain of mod$par.X after removing the first warmup iterations per chain. Has to be smaller than nrow(mod$par.X[[1]]) minus warmup.
warmup	The number of iterations discarded for warmup per chain. The default is NULL in which case, conservatively, the first half of posterior samples is discarded.
cores	Optional integer; number of cores used for parallel evaluation of the observation-level likelihood contributions. Defaults to 2. To set to maximum available cores set to NULL.

Details

See get_IC(). This function does not do parallel computation using foreach and instead uses apply. This may be useful on machines without foreach support or when parallel processing is not desired since foreach is used in a higher level environment (e.g. Monte Carlo simulation).

Value

A numeric matrix with one row and three columns:

`WAIC1` The first WAIC estimate based on the mean log pointwise posterior density and a bias correction using the average log likelihood.

`WAIC2` The second WAIC estimate based on the mean log pointwise posterior density and a bias correction using the posterior variance of the log likelihood contributions.

`DIC` The deviance information criterion.

See Also

`bayestsm()` `bayestsm_seq()`

---

mod_slice *A small fitted 'bayestsm' model for examples*

Description

A `bayestsm` model object fitted to a small simulated screening data set. It is shipped with the package so that documentation examples for downstream functions (`get_IC`, `ppCIF`, `plot.ppCIF`) run quickly and reproducibly without re-fitting.

Usage

```
mod_slice
```

Format

An object of class `"bayestsm"` as returned by `bayestsm()`; see the **Value** section of `bayestsm()` for the list structure.

Details

Generated with `set.seed(1)` from a two-covariate Weibull/Weibull model on `n = 1000` observations using `mc = 100` slice-sampler iterations and `chains = 2`. The number of draws is deliberately small and intended only to illustrate the package API, not for inference.

Examples

```
data(mod_slice)
summary(mod_slice)
```

plot.bayestsm *Plot method for bayestsm objects*

Description

Plot method for bayestsm objects

Usage

```
## S3 method for class 'bayestsm'
plot(x, warmup = 0, plot.X = TRUE, plot.S = TRUE, ...)
```

Arguments

x	An object of class "bayestsm".
warmup	Number of initial MCMC iterations to discard.
plot.X	Logical; whether trace plots for the parameters of the x-model should be made.
plot.S	Logical; whether trace plots for the parameters of the s-model should be made.
...	Additional arguments passed to plotting functions.

Value

No return value, called for its side effect of printing MCMC trace plots for the parameters of the x-model and/or s-model.

plot.ppCIF *Plot posterior predictive CIFs*

Description

Plot method for objects returned by ppCIF().

Usage

```
## S3 method for class 'ppCIF'
plot(
  x,
  y = NULL,
  ci = TRUE,
  layout = c("separate", "combined"),
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  facet_scales = "free_x",
  ...
)
```

Arguments

x	An object returned by ppCIF().
y	Ignored.
ci	Logical. If TRUE, pointwise 95 percent credible regions are added as shaded areas.
layout	Character string. Either "separate" for three horizontally adjacent panels, or "combined" for a single chart with all three curves.
main	Optional plot title.
xlab	Optional x-axis label.
ylab	Optional y-axis label.
xlim	Optional numeric vector of length two giving the x-axis limits. If NULL, the limits are chosen automatically.
facet_scales	Character string passed to ggplot2::facet_wrap(scales = ...) when layout = "separate". The default is "free_x".
...	Further graphical arguments, currently ignored.

Details

Produces posterior predictive CIF plots for X, S, and Y = X + S. Regardless of whether the object was created with type = "quantiles" or type = "percentiles", the plot is shown in CDF form with time on the x-axis and probability on the y-axis.

If x$type = "quantiles", the stored posterior predictive CDF values are plotted directly. If x$type = "percentiles", the stored posterior predictive quantiles are plotted against the probability grid, so that the resulting graph is again displayed in CDF form.

Value

Invisibly returns the resulting ggplot object.

See Also

bayestsm() ppCIF()

---

ppCIF *Posterior Predictive CIFs for X, S, and $Y = X + S$*

Description

Computes posterior predictive cumulative incidence functions (CIFs) for the latent event times X, S, and $Y = X + S$. In the present progressive three-state setting, these CIFs are identical to the posterior predictive cumulative distribution functions (CDFs) of the corresponding latent times.

Usage

```
ppCIF(
  mod,
  fix_Z.X = NULL,
  fix_Z.S = NULL,
  pst.samples = 1000,
  warmup = NULL,
  perc = seq(0, 0.99, 0.01),
  quant = NULL,
  method = c("simulation", "analytic"),
  type = c("percentiles", "quantiles"),
  cred_int = c(0.025, 0.975)
)
```

Arguments

mod	A fitted bayestsm model object.
fix_Z.X	Optional numeric vector used to fix covariates in the X-submodel. Its length must equal ncol(mod$Z.X). Entries correspond to the columns of mod$Z.X. Values set to NA mean that the corresponding column (variable) in Z.X is not fixed and, thus, empirically marginalized out. A valued entry specified by the user means that the CIF X is estimated conditional on that value.
fix_Z.S	Optional numeric vector used to fix covariates in the S-submodel. See fix_Z.X.
pst.samples	Posterior samples used to compute posterior predictive CIFs.
warmup	The number of iterations discarded for warmup per chain. The default is NULL in which case, conservatively, the first half of posterior samples is discarded.
perc	Numeric vector of probabilities in $[0, 1)$ used when type = "percentiles". For each posterior draw, posterior predictive quantiles are evaluated at these probabilities.
quant	Optional numeric vector of time points used when type = "quantiles". At these values, the posterior predictive CDF is evaluated. If NULL, a grid is constructed automatically from 0 to the largest finite follow-up time in the data.
method	Character string specifying the computational method. Currently supported values are "analytic" and "simulation". See details.
type	Character string specifying whether posterior predictive inference is returned as CDF values on a time grid (type = "quantiles") or as quantiles on a probability grid (type = "percentiles"). The latter is available only with method = "simulation".
cred_int	The vector of the lower and upper bound of the credible (posterior) intervals returned. Default (2.5%, 97.5%): c(0.025, 0.975).

Details

Let $\theta^{(m)}$ denote posterior draw $m = 1, \ldots, M$ and let z_i be the covariate vector for individual $i = 1, \ldots, n$. For a given latent variable $T \in \{X, S\}$, the posterior predictive CDF at covariate pattern z_i is

$$F_T(t \mid z_i, \theta^{(m)}) = P(T \leq t \mid z_i, \theta^{(m)}).$$

If covariates are not fully fixed, the function returns a covariate-marginal posterior predictive CDF obtained by averaging over the empirical distribution of the covariates in the fitted sample:

$$\bar{F}_T(t \mid \theta^{(m)}) = \frac{1}{n}\sum_{i=1}^{n} F_T(t \mid z_i, \theta^{(m)}).$$

This yields a posterior sample of predictive CIFs, from which pointwise posterior medians and credible intervals are reported.

Two computational methods (`method`) are available which in practice give very similar results, provided `pst.samples` is chosen large.

With `method = "analytic"`, closed-form evaluation of the CDF is used whenever available. For $T \in \{X, S\}$ and a grid of time points $q_1, \ldots, q_K$, this computes

$$\bar{F}_T(q_k \mid \theta^{(m)}) = \frac{1}{n}\sum_{i=1}^{n} F_T(q_k \mid z_i, \theta^{(m)}).$$

For $Y = X + S$, no closed-form expression is available in general, so its posterior predictive CDF is still approximated by Monte Carlo simulation under `method = "analytic"`.

With `method = "simulation"`, posterior predictive samples are generated for all requested quantities. For each posterior draw $\theta^{(m)}$, draws

$$X_i^{(m)} \sim p(X \mid z_i, \theta^{(m)}), \qquad S_i^{(m)} \sim p(S \mid z_i, \theta^{(m)}), \qquad Y_i^{(m)} = X_i^{(m)} + S_i^{(m)}$$

are generated and empirical CDFs or empirical quantiles are computed from these simulated samples.

The argument `type` determines whether inference is returned on a grid of time points or on a grid of probabilities.

If `type = "quantiles"`, the function evaluates the posterior predictive CDF on the supplied grid `quant = (q_1,...,q_K)` and returns

$$\bar{F}_T(q_k \mid \theta^{(m)}), \qquad k = 1, \ldots, K.$$

Thus, despite the name, `type = "quantiles"` means that the grid consists of quantiles or time points at which the CDF is evaluated.

If `type = "percentiles"`, the function evaluates the posterior predictive quantile function on the supplied probability grid `perc = (p_1,...,p_K)` and returns

$$Q_T(p_k \mid \theta^{(m)}) = \inf\{t : F_T(t \mid \theta^{(m)}) \geq p_k\}, \qquad k = 1, \ldots, K.$$

This option is available only with `method = "simulation"`.

The arguments `fix_Z.X` and `fix_Z.S` allow selective fixing of covariates when computing posterior predictive CIFs. These vectors must have the same lengths as the numbers of columns in `mod$Z.X` and `mod$Z.S`, respectively. Entries equal to `NA` indicate that the corresponding covariate should remain unchanged and therefore still be averaged over empirically. Non-missing entries replace the corresponding covariate values for all individuals before computing the predictive CIF.

This makes it possible to obtain subgroup-specific posterior predictive CIFs. For example, a specific age value can be fixed while all remaining covariates are left at `NA`, yielding a posterior predictive

CIF for that age group while marginalizing over the empirical distribution of the other covariates. Formally, if the covariate vector is partitioned into fixed components z^{fix} and unfixed components z_i^{free}, the function computes

$$\bar{F}_T(t \mid z^{\text{fix}}, \theta^{(m)}) = \frac{1}{n} \sum_{i=1}^{n} F_T\left(t \mid z^{\text{fix}}, z_i^{\text{free}}, \theta^{(m)}\right).$$

If all covariates are fixed, no marginalization over covariates remains and a fully conditional posterior predictive CIF is obtained:

$$F_T(t \mid z^*, \theta^{(m)}).$$

Value

A list with components depending on `type`.

If `type = "quantiles"`, the returned list contains:

`med.p.x`, `med.p.s`, `med.p.y` Pointwise posterior medians of the predictive CDFs of X, S, and $Y = X + S$ on the grid `grid`.

`p.x.ci`, `p.s.ci`, `p.y.ci` Pointwise 95\ posterior credible intervals for the predictive CDFs.

`grid` The grid of time points at which the predictive CDFs were evaluated.

`type` The supplied `type` argument.

If `type = "percentiles"`, the returned list contains:

`med.q.x`, `med.q.s`, `med.q.y` Pointwise posterior medians of the predictive quantile functions of X, S, and $Y = X + S$ on the probability grid `grid`.

`q.x.ci`, `q.s.ci`, `q.y.ci` Pointwise 95\ posterior credible intervals for the predictive quantile functions.

`grid` The grid of probabilities at which the predictive quantile functions were evaluated.

`type` The supplied `type` argument.

See Also

`bayestsm()` `plot.ppCIF()`

Examples

```
data(mod_slice) # A toy example model with only 100 posterior draws


# Obtain quantiles for provided percentiles
# In practice choose larger number of pst.samples for reliable estimation
postCDF_perc = ppCIF(mod_slice,
                     pst.samples = 50,
                     perc = seq(0, 0.99, 0.01),
                     method = 'simulation',
                     type = 'percentiles')

# straight forward plot over provided percentiles - quantiles combintation
```

```
plot(postCDF_perc, xlim = c(0,40))

# Obtain percentiles for provided quantiles (default: grid between 0 and max. follow up)
# In practice choose larger number of pst.samples for reliable estimation
postCDF_quant = ppCIF(mod_slice,
                      pst.samples = 50,
                      method = 'simulation',
                      type = 'quantiles')

# plot similar to plot of postCDF_perc
plot(postCDF_quant, xlim = c(0,40))

# Obtain percentiles for specific quantiles (e.g. probability for transition by 5 time units)
(postCDF_quant2 = ppCIF(mod_slice,
                        pst.samples = 50,
                        quant = c(5, 10),
                        method = 'simulation',
                        type = 'quantiles'))

# Alternatively, method = analytic can be used for quantiles with very similar results
postCDF_quant3 = ppCIF(mod_slice,
                       pst.samples = 50,
                       method = 'analytic',
                       type = 'quantiles')

plot(postCDF_quant3, xlim = c(0,40))
```

`search_prop_sd` *Heuristic search for a Metropolis-Hastings proposal standard deviation*

Description

Tunes the Metropolis-Hastings proposal standard deviation by iteratively extending a bayestsm fitted model run and adjusting the proposal scale until the observed acceptance rate falls within a target interval for a specified number of successive iterations.

Usage

```
search_prop_sd(
  m,
  mc = 3000,
  succ.min = 3,
  acc_bounds = c(0.21, 0.24),
  silent = FALSE
)
```

Arguments

m	A fitted model object containing acceptance information and current proposal settings. This object is updated iteratively during the search.
mc	Integer; number of additional MCMC iterations used at each tuning step. Defaults to 3000.
succ.min	Integer; number of successive successful tuning iterations required before the search stops. Defaults to 3.
acc_bounds	Numeric vector of length 2 giving the lower and upper target bounds for the acceptance rate of the X parameter update. Defaults to c(0.2, 0.3).
silent	When output should be produced (FALSE) or not (TRUE).

Details

The function implements a simple stepwise heuristic for proposal tuning. It starts from the current proposal standard deviation stored in m$prop.sd and compares the observed Metropolis-Hastings acceptance rate with the target interval acc_bounds.

If the acceptance rate is below the lower target bound, the proposal standard deviation is decreased. If the acceptance rate is above the upper target bound, the proposal standard deviation is increased. Once the acceptance rate falls within the target interval, the current tuning step is counted as a success.

After each successful step, the number of MCMC iterations used for the next update is doubled. The search stops after succ.min successive successful steps. This can help stabilize the tuning decision by checking the acceptance behaviour over progressively longer runs.

The function updates the model by calling bayestsm() with prev.run = m. As currently implemented, only the proposal standard deviation for the X update is tuned.

This is a heuristic tuning tool intended to provide a reasonable proposal scale before longer production runs. It does not guarantee optimal mixing or efficiency.

Value

A list with the following components:

prop.sd The tuned proposal standard deviation for the X update.

ac The final acceptance rate used for the stopping decision.

---

search_prop_sd_seq *Heuristic search for a Metropolis-Hastings proposal standard deviation (sequential processing)*

Description

Tunes the Metropolis-Hastings proposal standard deviation by iteratively extending a bayestsm fitted model run and adjusting the proposal scale until the observed acceptance rate falls within a target interval for a specified number of successive iterations.

Usage

```
search_prop_sd_seq(
  m,
  mc = 3000,
  succ.min = 3,
  acc_bounds = c(0.21, 0.24),
  silent = FALSE
)
```

Arguments

m	A fitted model object containing acceptance information and current proposal settings. This object is updated iteratively during the search.
mc	Integer; number of additional MCMC iterations used at each tuning step. Defaults to 3000.
succ.min	Integer; number of successive successful tuning iterations required before the search stops. Defaults to 3.
acc_bounds	Numeric vector of length 2 giving the lower and upper target bounds for the acceptance rate of the X parameter update. Defaults to c(0.2, 0.3).
silent	When output should be produced (FALSE) or not (TRUE).

Details

Like search_prop_sd() but instead of parallel processing through bayestsm sequential processing with a for loop through bayestsm_seq is done. This can be helpful in Monte Carlo simulation studies where parallelization within bayestsm is not desirable.

Value

A list with the following components:

prop.sd.X The tuned proposal standard deviation for the X update.

ac.X The final acceptance rate used for the stopping decision.

---

summary.bayestsm *Summary method for bayestsm objects*

Description

Summary method for bayestsm objects

Usage

```
## S3 method for class 'bayestsm'
summary(object, warmup = 0, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

object	An object of class "bayestsm".
warmup	Number of initial MCMC iterations to discard.
probs	Numeric vector of posterior quantiles.
...	Additional arguments, currently unused.

Value

Invisibly returns a list of class "summary.bayestsm" with the following components:

table.X A data frame with posterior quantiles and convergence diagnostics ($\hat{R}$ and ESS) for the parameters of the x-model.

table.S A data frame with posterior quantiles and convergence diagnostics ($\hat{R}$ and ESS) for the parameters of the s-model.

convergence.criteria A character string summarising the convergence criteria used.

draws.info A character string reporting the total number of posterior draws saved after thinning and the total draws before thinning.

The function is also called for its side effect of printing the above information to the console.

---

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 = 1, end = nrow(as.matrix(obj[1])), thinning = 1)
```

Arguments

obj	An object of class mcmc.list.
burnin	Integer; first iteration to retain. Defaults to 1.
end	Integer; last iteration to retain. Defaults to the number of rows in the first chain.
thinning	Integer; thinning interval. Defaults to 1.

Details

The function subsets each chain in obj to the iterations seq(burnin, end, thinning) and reconstructs the result as an mcmc.list.

Note that the argument name is thinning to match the existing function definition.

Value

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

Index

∗ **datasets**
mod_slice, 18

bayestsm, 2, *25, 26*
bayestsm(), *11*, *16–18*, *20*, *23*
bayestsm_seq, 8, *26*
bayestsm_seq(), *18*

gendat, 13
get_IC, 16, *18*
get_IC(), *17*
get_IC_seq, 17

log_aft_prior(), *5–7*, *11*, *13*

mod_slice, 18

plot.bayestsm, 19
plot.bayestsm(), *7*, *13*
plot.ppCIF, *18*, 19
plot.ppCIF(), *23*
ppCIF, *18*, 20
ppCIF(), *7*, *13*, *20*

search_prop_sd, 24
search_prop_sd(), *7*, *13*, *26*
search_prop_sd_seq, 25
summary.bayestsm, 26
summary.bayestsm(), *7*, *13*

trim.mcmc, 27
trim.mcmc(), *7*, *13*