Package ‘bayesCureRateModel’

June 18, 2025

Type Package

Title Bayesian Cure Rate Modeling for Time-to-Event Data

Version 1.4

Date 2025-06-18

Maintainer Panagiotis Papastamoulis <papapast@yahoo.gr>

Description A fully Bayesian approach in order to estimate a general family of cure rate models under the presence of covariates, see Papastamoulis and Milienos (2024) <doi:10.1007/s11749-024-00942-w> and Papastamoulis and Milienos (2024b) <doi:10.48550/arXiv.2409.10221>. The promotion time can be modelled (a) parametrically using typical distributional assumptions for time to event data (including the Weibull, Exponential, Gompertz, log-Logistic distributions), or (b) semiparametrically using finite mixtures of distributions. In both cases, user-defined families of distributions are allowed under some specific requirements. Posterior inference is carried out by constructing a Metropolis-coupled Markov chain Monte Carlo (MCMC) sampler, which combines Gibbs sampling for the latent cure indicators and Metropolis-Hastings steps with Langevin diffusion dynamics for parameter updates. The main MCMC algorithm is embedded within a parallel tempering scheme by considering heated versions of the target posterior distribution.

License GPL-2

URL https://github.com/mqbssppe/Bayesian_cure_rate_model

Imports Rcpp (>= 1.0.12), survival, doParallel, parallel, foreach, mclust, coda, HDInterval, VGAM, calculus, flexsurv

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Panagiotis Papastamoulis [aut, cre] (<https://orcid.org/0000-0001-9468-7613>), Fotios Milienos [aut] (<https://orcid.org/0000-0003-1423-7132>)

Depends R (>= 3.5.0)

R topics documented:

`bayesCureRateModel-package`
Bayesian Cure Rate Modeling for Time-to-Event Data

Description

A fully Bayesian approach in order to estimate a general family of cure rate models under the presence of covariates, see Papastamoulis and Milienos (2024) <doi:10.1007/s11749-024-00942-w> and Papastamoulis and Milienos (2024b) <doi:10.48550/arXiv.2409.10221>. The promotion time can be modelled (a) parametrically using typical distributional assumptions for time to event data (including the Weibull, Exponential, Gompertz, log-Logistic distributions), or (b) semiparametrically using finite mixtures of distributions. In both cases, user-defined families of distributions are allowed under some specific requirements. Posterior inference is carried out by constructing a Metropolis-coupled Markov chain Monte Carlo (MCMC) sampler, which combines Gibbs sampling for the latent cure indicators and Metropolis-Hastings steps with Langevin diffusion dynamics for parameter updates. The main MCMC algorithm is embedded within a parallel tempering scheme by considering heated versions of the target posterior distribution.

The main function of the package is `cure_rate_MC3`. See details for a brief description of the model.

Details

Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ denote the observed data, which correspond to time-to-event data or censoring times. Let also $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{x_{ip}})'$ denote the covariates for subject i, $i = 1, \ldots, n$.

Assuming that the n observations are independent, the observed likelihood is defined as

$$L = L(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{x}) = \prod_{i=1}^{n} f_P(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i)^{\delta_i} S_P(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i)^{1-\delta_i},$$

where $\delta_i = 1$ if the i-th observation corresponds to time-to-event while $\delta_i = 0$ indicates censoring time. The parameter vector $\boldsymbol{\theta}$ is decomposed as

$$\boldsymbol{\theta} = (\boldsymbol{\alpha}', \boldsymbol{\beta}', \gamma, \lambda)$$

where

- $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)' \in \mathcal{A}$ are the parameters of the promotion time distribution whose cumulative distribution and density functions are denoted as $F(\cdot, \boldsymbol{\alpha})$ and $f(\cdot, \boldsymbol{\alpha})$, respectively.
- $\boldsymbol{\beta} \in \mathbf{R}^k$ are the regression coefficients with k denoting the number of columns in the design matrix (it may include a constant term or not).
- $\gamma \in \mathbf{R}$
- $\lambda > 0$.

The population survival and density functions are defined as

$$S_P(y; \boldsymbol{\theta}) = \left(1 + \gamma \exp\{\boldsymbol{x}_i \boldsymbol{\beta}'\} c^{\gamma \exp\{\boldsymbol{x}_i \boldsymbol{\beta}'\}} F(y; \boldsymbol{\alpha})^{\lambda}\right)^{-1/\gamma}$$

whereas,

$$f_P(y; \boldsymbol{\theta}) = -\frac{\partial S_P(y; \boldsymbol{\theta})}{\partial y}.$$

Finally, the cure rate is affected through covariates and parameters as follows

$$p_0(\boldsymbol{x}_i; \boldsymbol{\theta}) = \left(1 + \gamma \exp\{\boldsymbol{x}_i \boldsymbol{\beta}'\} c^{\gamma \exp\{\boldsymbol{x}_i \boldsymbol{\beta}'\}}\right)^{-1/\gamma}$$

where $c = e^{e^{-1}}$.

The promotion time distribution can be a member of standard families (currently available are the following: Exponential, Weibull, Gamma, Lomax, Gompertz, log-Logistic) and in this case $\alpha = (\alpha_1, \alpha_2) \in (0, \infty)^2$. Also considered is the Dagum distribution, which has three parameters $(\alpha_1, \alpha_2, \alpha_3) \in (0, \infty)^3$. In case that the previous parametric assumptions are not justified, the promotion time can belong to the more flexible family of finite mixtures of Gamma distributions. For example, assume a mixture of two Gamma distributions of the form

$$f(y; \boldsymbol{\alpha}) = \alpha_5 f_{\mathcal{G}}(y; \alpha_1, \alpha_3) + (1 - \alpha_5) f_{\mathcal{G}}(y; \alpha_2, \alpha_4),$$

where

$$f_{\mathcal{G}}(y; \alpha, \beta) = \frac{\beta^{\alpha}}{\Gamma(\alpha)} y^{\alpha-1} \exp\{-\beta y\}, y > 0$$

denotes the density of the Gamma distribution with parameters $\alpha > 0$ (shape) and $\beta > 0$ (rate). For the previous model, the parameter vector is

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)' \in \mathcal{A}$$

where $\mathcal{A} = (0, \infty)^4 \times (0, 1)$.

User defined promotion time distributions and finite mixtures of them can be also fitted using the options `'user'` and `'user_mixture'`, respectively. The appropriate model can be selected according to information criteria such as the BIC.

The binary vector $\boldsymbol{I} = (I_1, \ldots, I_n)$ contains the (latent) cure indicators, that is, $I_i = 1$ if the i-th subject is susceptible and $I_i = 0$ if the i-th subject is cured. Δ_0 denotes the subset of $\{1, \ldots, n\}$ containing the censored subjects, whereas $\Delta_1 = \Delta_0^c$ is the (complementary) subset of uncensored subjects. The complete likelihood of the model is

$$L_c(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{I}) = \prod_{i \in \Delta_1} (1 - p_0(\boldsymbol{x}_i, \boldsymbol{\theta})) f_U(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i) \prod_{i \in \Delta_0} p_0(\boldsymbol{x}_i, \boldsymbol{\theta})^{1 - I_i} \{(1 - p_0(\boldsymbol{x}_i, \boldsymbol{\theta})) S_U(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i)\}^{I_i}.$$

f_U and S_U denote the probability density and survival function of the susceptibles, respectively, that is

$$S_U(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i) = \frac{S_P(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i) - p_0(\boldsymbol{x}_i; \boldsymbol{\theta})}{1 - p_0(\boldsymbol{x}_i; \boldsymbol{\theta})}, f_U(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i) = \frac{f_P(y_i; \boldsymbol{\theta}, \boldsymbol{x}_i)}{1 - p_0(\boldsymbol{x}_i; \boldsymbol{\theta})}.$$

Index of help topics:

```
Surv                    Create a Survival Object
bayesCureRateModel-package
                        Bayesian Cure Rate Modeling for Time-to-Event
                        Data
complete_log_likelihood_general
                        Logarithm of the complete log-likelihood for
                        the general cure rate model.
compute_fdr_tpr         Compute the achieved FDR and TPR
cure_rate_MC3           Main function of the package
cure_rate_mcmc          The basic MCMC scheme.
logLik.bayesCureModel   Extract the log-likelihood.
log_dagum               PDF and CDF of the Dagum distribution
log_gamma               PDF and CDF of the Gamma distribution
log_gamma_mixture       PDF and CDF of a Gamma mixture distribution
log_gompertz            PDF and CDF of the Gompertz distribution
log_logLogistic         PDF and CDF of the log-Logistic distribution.
log_lomax               PDF and CDF of the Lomax distribution
log_user_mixture        Define a finite mixture of a given family of
                        distributions.
log_weibull             PDF and CDF of the Weibull distribution
marriage_dataset        Marriage data
plot.bayesCureModel     Plot method
plot.predict_bayesCureModel
                        Plot method
predict.bayesCureModel
```

```
                         Predict method.
print.bayesCureModel     Print method
print.predict_bayesCureModel
                         Print method for the 'predict' object
print.summary_bayesCureModel
                         Print method for the summary
residuals.bayesCureModel
                         Computation of residuals.
sim_mix_data             Simulated dataset
summary.bayesCureModel
                         Summary method.
summary.predict_bayesCureModel
                         Summary method for predictions.
```

Author(s)

Panagiotis Papastamoulis and Fotios S. Milienos

Maintainer: Panagiotis Papastamoulis <papapast@yahoo.gr>

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

Papastamoulis and Milienos (2024). bayesCureRateModel: Bayesian Cure Rate Modeling for Time to Event Data in R. arXiv:2409.10221

See Also

cure_rate_MC3

Examples

```
# TOY EXAMPLE (very small numbers... only for CRAN check purposes)
# simulate toy data
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2,
  data = my_data_frame,
  promotion_time = list(family = 'weibull'),
  nChains = 2,
```

```
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
# print method
 fit1
#  summary method
 summary1 <- summary(fit1)

# WARNING: the following parameters
#  mcmc_cycles, nChains
#       should take _larger_ values. E.g. a typical implementation consists of:
#       mcmc_cycles = 15000, nChains = 12
```

`complete_log_likelihood_general` *Logarithm of the complete log-likelihood for the general cure rate model.*

Description

Compute the logarithm of the complete likelihood, given a realization of the latent binary vector of cure indicators `I_sim` and current values of the model parameters `g`, `lambda`, `b` and promotion time parameters (α) which yield log-density values (one per observation) stored to the vector `log_f` and log-cdf values stored to the vector `log_F`.

Usage

```
complete_log_likelihood_general(y, X, Censoring_status,
 g, lambda, log_f, log_F, b, I_sim, alpha)
```

Arguments

`y`	observed data (time-to-event or censored time)
`X`	design matrix. Should contain a column of 1's if the model has a constant term.
`Censoring_status`	binary variables corresponding to time-to-event and censoring.
`g`	The γ parameter of the model (real).
`lambda`	The λ parameter of the model (positive).
`log_f`	A vector containing the natural logarithm of the density function of the promotion time distribution per observation, for the current set of parameters. Its length should be equal to the sample size.
`log_F`	A vector containing the natural logarithm of the cumulative density function of the promotion time distribution per observation, for the current set of parameters. Its length should be equal to the sample size.
`b`	Vector of regression coefficients. Its length should be equal to the number of columns of the design matrix.

`I_sim` Binary vector of the current value of the latent cured status per observation. Its length should be equal to the sample size. A time-to-event cannot be cured.

`alpha` A parameter between 0 and 1, corresponding to the temperature of the complete posterior distribution.

Details

The complete likelihood of the model is

$$L_c(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{I}) = \prod_{i\in\Delta_1}(1-p_0(\boldsymbol{x}_i,\boldsymbol{\theta}))f_U(y_i;\boldsymbol{\theta},\boldsymbol{x}_i)\prod_{i\in\Delta_0}p_0(\boldsymbol{x}_i,\boldsymbol{\theta})^{1-I_i}\{(1-p_0(\boldsymbol{x}_i,\boldsymbol{\theta}))S_U(y_i;\boldsymbol{\theta},\boldsymbol{x}_i)\}^{I_i}.$$

f_U and S_U denote the probability density and survival function of the susceptibles, respectively, that is

$$S_U(y_i;\boldsymbol{\theta},\boldsymbol{x}_i) = \frac{S_P(y_i;\boldsymbol{\theta},\boldsymbol{x}_i) - p_0(\boldsymbol{x}_i;\boldsymbol{\theta})}{1-p_0(\boldsymbol{x}_i;\boldsymbol{\theta})}, f_U(y_i;\boldsymbol{\theta},\boldsymbol{x}_i) = \frac{f_P(y_i;\boldsymbol{\theta},\boldsymbol{x}_i)}{1-p_0(\boldsymbol{x}_i;\boldsymbol{\theta})}.$$

Value

A list with the following entries

`cll` the complete log-likelihood for the current parameter values.

`logS` Vector of logS values (one for each observation).

`logP0` Vector of logP0 values (one for each observation).

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2023). Bayesian inference and cure rate modeling for event history data. arXiv:2310.06926.

Examples

```
# simulate toy data
 set.seed(1)
 n = 4
 stat = rbinom(n, size = 1, prob = 0.5)
 x <- cbind(1, matrix(rnorm(n), n, 1))
 y <- rexp(n)
 lw <- log_weibull(y, a1 = 1, a2 = 1, c_under = 1e-9)
# compute complete log-likelihood
complete_log_likelihood_general(y = y, X = x,
 Censoring_status = stat,
 g = 1, lambda = 1,
 log_f = lw$log_f, log_F = lw$log_F,
 b = c(-0.5,0.5),
 I_sim = stat, alpha = 1)
```

compute_fdr_tpr *Compute the achieved FDR and TPR*

Description

This help function computes the achieved False Discovery Rate (FDR) and True Positive Rate (TPR). Useful for simulation studies where the ground truth classification of subjects in susceptibles and cured items is known.

Usage

```
compute_fdr_tpr(true_latent_status, posterior_probs,
 myCut = c(0.01, 0.05, 0.1, 0.15))
```

Arguments

`true_latent_status`
: a binary vector containing the true latent status: 1 should correspond to the positive instances ("cured") and 0 to the negative ("susceptibles").

`posterior_probs`
: a numeric vector with entries between 0 and 1 containing the scores (posterior probabilities) of being positive ("cured") for each item.

`myCut`
: Vector containing the desired nominal FDR levels.

Value

This function will return for every nominal FDR level the following quantities:

`achieved_fdr` the achieved false discovery rate.

`tpr` the true positive rate.

`nominal_fdr` the nominal FDR level.

Author(s)

Panagiotis Papastamoulis

Examples

```
set.seed(1)
v1 <- sample(0:1, size = 100, replace=TRUE, prob=c(0.8,0.2) )
v2 <- runif(100)
compute_fdr_tpr(true_latent_status = v1, posterior_probs = v2)
```

cure_rate_MC3 *Main function of the package*

Description

Runs a Metropolis Coupled MCMC (MC^3) sampler in order to estimate the joint posterior distribution of the model.

Usage

```
cure_rate_MC3(formula, data, nChains = 12, mcmc_cycles = 15000,
 alpha = NULL,nCores = 1, sweep = 5, a_g = 1, b_g = 1,
 a_l = 2.1, b_l = 1.1, mu_b = NULL,
 Sigma = NULL, g_prop_sd = 0.045,
 lambda_prop_scale = 0.03, b_prop_sd = NULL,
 initialValues = NULL, plot = TRUE, adjust_scales = FALSE,
 verbose = FALSE, tau_mala = 1.5e-05, mala = 0.15,
 promotion_time = list(family = "weibull",
 prior_parameters = matrix(rep(c(2.1, 1.1), 2), byrow = TRUE, 2, 2),
 prop_scale = c(0.1, 0.2)), single_MH_in_f = 0.2, c_under = 1e-9)
```

Arguments

formula	an object of class formula: a symbolic description of the model to be fitted. The left-hand side should be a Surv object, a class inherited from the **survival** package. The binary censoring indicators are interpreted as a time-to-event (1) or as a censoring time (0).
data	a data frame containing all variable names included in formula.
nChains	Positive integer corresponding to the number of heated chains in the MC^3 scheme.
mcmc_cycles	Length of the generated MCMC sample. Default value: 15000. Note that each MCMC cycle consists of sweep (see below) usual MCMC iterations.
alpha	A decreasing sequence in $[1, 0)$ of nChains temperatures (or heat values). The first value should always be equal to 1, which corresponds to the target posterior distribution (that is, the first chain).
nCores	The number of cores used for parallel processing. In case where nCores > 1, the nChains will be processed in parallel using nCores cores. This may speed up significantly the run-time of the algorithm in Linux or macOS, but it is not suggested in Windows.
sweep	The number of usual MCMC iterations per MCMC cycle. Default value: 10.
a_g	Parameter a_γ of the prior distribution of γ.
b_g	Parameter b_γ of the prior distribution of γ.
a_l	Shape parameter a_λ of the Inverse Gamma prior distribution of λ.
b_l	Scale parameter b_λ of the Inverse Gamma prior distribution of λ.

mu_b — Mean ($\boldsymbol{\mu}$) of the multivariate normal prior distribution of regression coefficients. Should be a vector whose length is equal to k, i.e. the number of columns of the design matrix X. Default value: rep(0, k).

Sigma — Covariance matrix of the multivariate normal prior distribution of regression coefficients.

g_prop_sd — The scale of the proposal distribution for single-site updates of the γ parameter.

lambda_prop_scale — The scale of the proposal distribution for single-site updates of the λ parameter.

b_prop_sd — The scale of the proposal distribution for the update of the β parameter (regression coefficients).

initialValues — A list of initial values for each parameter (optional).

plot — Plot MCMC sample on the run. Default: TRUE.

adjust_scales — Boolean. If TRUE the MCMC sampler runs an initial phase of a small number of iterations in order to tune the scale of the proposal distributions in the Metropolis-Hastings steps.

verbose — Print progress on the terminal if TRUE.

tau_mala — Scale of the Metropolis adjusted Langevin diffussion proposal distribution.

mala — A number between $[0, 1]$ indicating the proportion of times the sampler attempts a MALA proposal. Thus, the probability of attempting a typical Metropolis-Hastings move is equal to 1 - mala.

promotion_time — A list with details indicating the parametric family of distribution describing the promotion time and corresponding prior distributions. See 'details'.

single_MH_in_f — The probability for attempting a series of single site updates in the typical Metropolis-Hastings move. Otherwise, with probability 1 - single_MH_in_f a Metropolis-Hastings move will attempt to update all continuous parameters simultaneously. It only makes sense when mala < 1.

c_under — A small positive number (much smaller than 1) which is used as a threshold in the CDF of the promotion time for avoiding underflows in the computation of the log-likelihood function. Default value: 1e-9.

Details

It is advised to scale all continuous explanatory variables in the design matrix, so their sample mean and standard deviations are equal to 0 and 1, respectively. No missing or infinite values are accepted. The promotion_time argument should be a list containing the following entries

family Character string specifying the family of distributions $\{F(\cdot;\boldsymbol{\alpha});\boldsymbol{\alpha} \in \mathcal{A}\}$ describing the promotion time.

prior_parameters Values of hyper-parameters in the prior distribution of the parameters $\boldsymbol{\alpha}$.

prop_scale The scale of the proposal distributions for each parameter in $\boldsymbol{\alpha}$.

K Optional. The number of mixture components in case where a mixture model is fitted, that is, when setting distribution to either 'gamma_mixture' or 'user_mixture'.

dirichlet_concentration_parameter Optional. Relevant only in the case of the 'gamma_mixture' or 'user_mixture'. Positive scalar (typically, set to 1) determining the (common) concentration parameter of the Dirichlet prior distribution of mixing proportions.

The family specifies the distributional family of promotion time and corresponds to a character string with available choices: 'exponential', 'weibull', 'gamma', 'logLogistic', 'gompertz', 'lomax', 'dagum', 'gamma_mixture'. User defined promotion time distributions and finite mixtures of them can be also fitted using the options family = 'user' and family = 'user_mixture', respectively. See the 'Note' below for further details.

The joint prior distribution of $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$ factorizes into products of inverse Gamma distributions for all (positive) parameters of F. Moreover, in the case of 'gamma_mixture' or 'user_mixture', the joint prior also consists of another term to the Dirichlet prior distribution on the mixing proportions.

The prop_scale argument should be a vector with length equal to the length of vector d (number of elements in $\boldsymbol{\alpha}$), containing (positive) numbers which correspond to the scale of the proposal distribution. Note that these scale parameters are used only as initial values in case where adjust_scales = TRUE.

Value

An object of class bayesCureModel, i.e. a list with the following entries

mcmc_sample
: Object of class mcmc (see the **coda** package), containing the generated MCMC sample for the target chain. The column names correspond to

  g_mcmc Sampled values for parameter γ

  lambda_mcmc Sampled values for parameter λ

  alpha1_mcmc**...** Sampled values for parameter α_1... of the promotion time distribution $F(\cdot; \alpha_1, \ldots, \alpha_d)$. The subsequent $d-1$ columns contain the sampled values for all remaining parameters, $\alpha_2, ..., \ldots, \alpha_d$, where d depens on the family used in promotion_time.

  b0_mcmc Sampled values for the constant term of the regression (present only in the case where the design matrix X contains a column of 1s).

  b1_mcmc**...** Sampled values for the regression coefficient for the first explanatory variable, and similar for all subsequent columns.

latent_status_censored
: A data frame with the simulated binary latent status for each censored item.

complete_log_likelihood
: The complete log-likelihood for the target chain.

swap_accept_rate
: the acceptance rate of proposed swappings between adjacent MCMC chains.

all_cll_values
: The complete log-likelihood for all chains.

input_data_and_model_prior
: the input data, model specification and selected prior parameters values.

log_posterior
: the logarithm of the (non-augmented) posterior distribution (after integrating the latent cured-status parameters out), up to a normalizing constant.

map_estimate
: The Maximum A Posterior estimate of parameters.

BIC
: Bayesian Information Criterion of the fitted model.

AIC
: Akaike Information Criterion of the fitted model.

residuals	The Cox-Snell residuals of the fitted model.
logLik	The maximum log-likelihood value for the target chain.
n_parameters	The number of parameters of the model.
initial_values	The initial values per chain.

Note

User-defined promotion time distributions and finite mixtures of them can be fitted using the options `family = 'user'` and `family = 'user_mixture'`, respectively, in the `promotion_time` argument. In this case, the user should specify the distributional family in a separate argument named `define` which is passed as an additional entry in the `promotion_time`. This function should accept two input arguments `y` and `a` corresponding to the observed data (vector of positive numbers) and the parameters of the distribution (vector of positives). Pay attention to the positivity requirement of the parameters: if this is not the case, the user should suitably reparameterize the distribution in terms of positive parameters. The `define` function should return in the form of a list two arguments: `log_f` and `log_F`, corresponding to the the logarithm of the probability density function and the logarithm of the cumulative density function, respectively, per observation. See Papastamoulis and Milienos (2024b) for examples of the implementation.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

Papastamoulis and Milienos (2024b). bayesCureRateModel: Bayesian Cure Rate Modeling for Time to Event Data in R. arXiv:2409.10221

Plummer M, Best N, Cowles K, Vines K (2006). "CODA: Convergence Diagnosis and Output Analysis for MCMC." R News, 6(1), 7-11.

Therneau T (2024). A Package for Survival Analysis in R. R package version 3.7-0, https://CRAN.R-project.org/package=survival.

See Also

`cure_rate_mcmc`

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
```

```
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2,
  data = my_data_frame,
  promotion_time = list(family = 'weibull'),
  nChains = 2, nCores = 1,
  mcmc_cycles = 3, sweep = 2)
```

---

cure_rate_mcmc *The basic MCMC scheme.*

Description

This is core MCMC function. The continuous parameters of the model are updated using (a) single-site Metropolis-Hastings steps and (b) a Metropolis adjusted Langevin diffusion step. The binary latent variables of the model (cured status per censored observation) are updated according to a Gibbs step. This function is embedded to the main function of the package cure_rate_MC3 which runs parallel tempered MCMC chains.

Usage

```
cure_rate_mcmc(y, X, Censoring_status, m, alpha = 1, mu_g = 1, s2_g = 1,
 a_l = 2.1, b_l = 1.1, promotion_time = list(family = "weibull",
 prior_parameters = matrix(rep(c(2.1, 1.1), 2), byrow = TRUE, 2, 2),
 prop_scale = c(0.2, 0.03)), mu_b = NULL, Sigma = NULL, g_prop_sd = 0.045,
 lambda_prop_scale = 0.03, b_prop_sd = NULL, initialValues = NULL,
 plot = FALSE, verbose = FALSE, tau_mala = 1.5e-05, mala = 0.15,
 single_MH_in_f = 0.5, c_under = 1e-9)
```

Arguments

y	observed data (time-to-event or censored time)
X	design matrix. Should contain a column of 1's if the model has a constant term.
Censoring_status	binary variables corresponding to time-to-event and censoring.
m	number of MCMC iterations.
alpha	A value between 0 and 1, corresponding to the temperature of the complete posterior distribution. The target posterior distribution corresponds to alpha = 1.
mu_g	Parameter a_γ of the prior distribution of γ.
s2_g	Parameter b_γ of the prior distribution of γ.
a_l	Shape parameter a_λ of the Inverse Gamma prior distribution of λ.
b_l	Scale parameter b_λ of the Inverse Gamma prior distribution of λ.

- `promotion_time` A list containing the specification of the promotion time distribution. See 'details'.
- `mu_b` Mean μ of the multivariate normal prior distribution of regression coefficients. Should be a vector whose length is equal to the number of columns of the design matrix `X`.
- `Sigma` Covariance matrix of the multivariate normal prior distribution of regression coefficients.
- `g_prop_sd` The scale of the proposal distribution for single-site updates of the γ parameter.
- `lambda_prop_scale` The scale of the proposal distribution for single-site updates of the λ parameter.
- `b_prop_sd` The scale of the proposal distribution for the update of the β parameter (regression coefficients).
- `initialValues` A list of initial values for each parameter (optional).
- `plot` Boolean for plotting on the run.
- `verbose` Boolean for printing progress on the run.
- `tau_mala` scale of the MALA proposal.
- `mala` Propability of attempting a MALA step. Otherwise, a simple MH move is attempted.
- `single_MH_in_f` Probability of attempting a single-site MH move in the basic Metropolis-Hastings step. Otherwise, a joint update is attempted.
- `c_under` A small positive number (much smaller than 1) which is used as a threshold in the CDF of the promotion time for avoiding underflows in the computation of the log-likelihood function. Default value: 1e-9.

Value

A list containing the following entries

- `mcmc_sample` The sampled MCMC values per parameter. See 'note'.
- `complete_log_likelihood` Logarithm of the complete likelihood per MCMC iteration.
- `acceptance_rates` The acceptance rate per move.
- `latent_status_censored` The MCMC sample of the latent status per censored observation.
- `log_prior_density` Logarithm of the prior density per MCMC iteration.

Note

In the case where the promotion time distribution is a mixture model, the mixing proportions $w_1, \ldots, w_K$ are reparameterized according to the following transformation

$$w_j = \frac{\rho_j}{\sum_{i=1}^{K} \rho_i}, j = 1, \ldots, K$$

where $\rho_i > 0$ for $i = 1, \ldots, K-1$ and $\rho_K = 1$. The sampler returns the parameters $\rho_1, \ldots, \rho_{K-1}$, not the mixing proportions.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

See Also

cure_rate_MC3

Examples

```
# simulate toy data just for cran-check purposes
        set.seed(1)
        n = 10
        stat = rbinom(n, size = 1, prob = 0.5)
        x <- cbind(1, matrix(rnorm(2*n), n, 2))
        y <- rexp(n)
# run a weibull model (including const. term)
# for m = 10 mcmc iterations
        fit1 <- cure_rate_mcmc(y = y, X = x, Censoring_status = stat,
               plot = FALSE,
                promotion_time = list(family = 'weibull',
                        prior_parameters = matrix(rep(c(2.1, 1.1), 2),
                                                byrow = TRUE, 2, 2),
                        prop_scale = c(0.1, 0.1)
                ),
                m = 10)
# the generated mcmc sampled values
 fit1$mcmc_sample
```

---

logLik.bayesCureModel *Extract the log-likelihood.*

Description

Method to extract the log-likelihood of a bayesCureModel object.

Usage

```
## S3 method for class 'bayesCureModel'
logLik(object, ...)
```

Arguments

object	An object of class bayesCureModel
...	ignored.

Value

The maximum (observed) log-likelihood value obtained across the MCMC run.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

See Also

cure_rate_MC3

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2,
  data = my_data_frame,
  promotion_time = list(family = 'exponential'),
  nChains = 2,
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
 ll <- logLik(fit1)
```

---

log_dagum *PDF and CDF of the Dagum distribution*

Description

The Dagum distribution as evaluated at the **VGAM** package.

Usage

```
log_dagum(y, a1, a2, a3, c_under = 1e-09)
```

Arguments

y	observed data
a1	scale parameter
a2	shape1.a parameter
a3	shape2.p parameter
c_under	A small positive value corresponding to the underflow threshold, e.g. c_under = 1e-9.

Details

The Dagum distribution is a special case of the 4-parameter generalized beta II distribution.

Value

A list containing the following entries

log_f	natural logarithm of the pdf, evaluated at each datapoint.
log_F	natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

References

Thomas W. Yee (2015). Vector Generalized Linear and Additive Models: With an Implementation in R. New York, USA: Springer.

See Also

ddagum

Examples

```
log_dagum(y = 1:10, a1 = 1, a2 = 1, a3 = 1, c_under = 1e-9)
```

---

log_gamma *PDF and CDF of the Gamma distribution*

Description

Computes the pdf and cdf of the Gamma distribution.

Usage

```
log_gamma(y, a1, a2, c_under = 1e-09)
```

Arguments

y	observed data
a1	shape parameter
a2	rate parameter
c_under	A small positive value corresponding to the underflow threshold, e.g. c_under = 1e-9.

Value

A list containing the following entries

log_f	natural logarithm of the pdf, evaluated at each datapoint.
log_F	natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

See Also

dgamma

Examples

```
log_gamma(y = 1:10, a1 = 1, a2 = 1, c_under = 1e-9)
```

---

log_gamma_mixture *PDF and CDF of a Gamma mixture distribution*

Description

Computes the logarithm of the probability density function and cumulative density function per observation for each observation under a Gamma mixture model.

Usage

```
log_gamma_mixture(y, a1, a2, p, c_under = 1e-09)
```

Arguments

y	observed data
a1	vector containing the shape parameters of each Gamma mixture component
a2	vector containing the rate parameters of each Gamma mixture component
p	vector of mixing proportions
c_under	threshold for underflows.

Value

A list containing the following entries

log_f natural logarithm of the pdf, evaluated at each datapoint.

log_F natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

Examples

```
y <- runif(10)
a1 <- c(1,2)
a2 <- c(1,1)
p <- c(0.9,0.1)
log_gamma_mixture(y, a1, a2, p)
```

---

log_gompertz *PDF and CDF of the Gompertz distribution*

Description

The Gompertz distribution as evaluated at the **flexsurv** package.

Usage

```
log_gompertz(y, a1, a2, c_under = 1e-09)
```

Arguments

y observed data

a1 shape parameter

a2 rate parameter

c_under A small positive value corresponding to the underflow threshold, e.g. c_under = 1e-9.

Value

A list containing the following entries

log_f natural logarithm of the pdf, evaluated at each datapoint.

log_F natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

References

Christopher Jackson (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

See Also

dgompertz

Examples

```
log_gompertz(y = 1:10, a1 = 1, a2 = 1, c_under = 1e-9)
```

---

log_logLogistic   *PDF and CDF of the log-Logistic distribution.*

Description

The log-Logistic distribution as evaluated at the **flexsurv** package.

Usage

```
log_logLogistic(y, a1, a2, c_under = 1e-09)
```

Arguments

y	observed data
a1	shape parameter
a2	scale parameter
c_under	A small positive value corresponding to the underflow threshold, e.g. c_under = 1e-9.

Details

The log-logistic distribution is the probability distribution of a random variable whose logarithm has a logistic distribution.

Value

A list containing the following entries

log_f	natural logarithm of the pdf, evaluated at each datapoint.
log_F	natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

References

Christopher Jackson (2016). flexsurv: A Platform for Parametric Survival Modeling in R. Journal of Statistical Software, 70(8), 1-33. doi:10.18637/jss.v070.i08

See Also

dllogis

Examples

```
log_logLogistic(y = 1:10, a1 = 1, a2 = 1, c_under = 1e-9)
```

---

log_lomax *PDF and CDF of the Lomax distribution*

Description

The Lomax distribution as evaluated at the **VGAM** package.

Usage

```
log_lomax(y, a1, a2, c_under = 1e-09)
```

Arguments

y	observed data
a1	scale parameter
a2	shape parameter
c_under	A small positive value corresponding to the underflow threshold, e.g. c_under = 1e-9.

Details

The Lomax distribution is a special case of the 4-parameter generalized beta II distribution.

Value

A list containing the following entries

log_f	natural logarithm of the pdf, evaluated at each datapoint.
log_F	natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

References

Thomas W. Yee (2015). Vector Generalized Linear and Additive Models: With an Implementation in R. New York, USA: Springer.

See Also

dlomax

Examples

```
log_lomax(y = 1:10, a1 = 1, a2 = 1, c_under = 1e-9)
```

---

log_user_mixture *Define a finite mixture of a given family of distributions.*

Description

This function computes the logarithm of the probability density function and cumulative density function per observation for each observation under a user-defined mixture of a given family of distributions. The parameters of the given family of distributions should belong to (0, inf).

Usage

```
log_user_mixture(user_f, y, a, p, c_under = 1e-09)
```

Arguments

user_f	a user defined function that returns the logarithm of a given probability density and the corresponding logarithm of the cumulative distribution function. These arguments should be returned in the form of a list with two entries: log_f and log_F, containing the logarithm of the pdf and cdf values of y, respectively, for a given set of parameter values.
y	observed data
a	a matrix where each column corresponds to component specific parameters and the columns to different components. All parameters should be positive. The number of columns should be the same with the number of mixture components.
p	vector of mixing proportions
c_under	threshold for underflows.

Value

A list containing the following entries

log_f	natural logarithm of the pdf, evaluated at each datapoint.
log_F	natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

Examples

```
# We will define a mixture of 2 exponentials distributions.
# First we pass the exponential distribution at user_f
user_f <- function(y, a){
 log_f <- dexp(y, rate = a, log = TRUE)
 log_F <- pexp(y, rate = a, log.p = TRUE)
 result <- vector('list', length = 2)
 names(result) <- c('log_f', 'log_F')
 result[["log_f"]] = log_f
 result[["log_F"]] = log_F
 return(result)
}
# simulate some date
y <- runif(10)
# Now compute the log of pdf and cdf for a mixture of K=2 exponentials
p <- c(0.9,0.1)
a <- matrix(c(0.1, 1.5), nrow = 1, ncol = 2)
log_user_mixture(user_f = user_f, y = y, a = a, p = p)
```

---

log_weibull *PDF and CDF of the Weibull distribution*

Description

Computes the log pdf and cdf of the weibull distribution.

Usage

```
log_weibull(y, a1, a2, c_under)
```

Arguments

y	observed data
a1	shape parameter
a2	rate parameter
c_under	A small positive value corresponding to the underflow threshold, e.g. c_under = 1e-9.

Value

A list containing the following entries

log_f	natural logarithm of the pdf, evaluated at each datapoint.
log_F	natural logarithm of the CDF, evaluated at each datapoint.

Author(s)

Panagiotis Papastamoulis

Examples

```
log_weibull(y = 1:10, a1 = 1, a2 = 1, c_under = 1e-9)
```

---

marriage_dataset *Marriage data*

Description

The variable of interest (`time`) corresponds to the duration (in years) of first marriage for 1500 individuals. The available covariates are:

`age` age of respondent (in years) at the time of first marriage. The values are standardized (sample mean and variance equal to 0 and 1, respectively).

`kids` factor: whether there were kids during the first marriage (`"yes"`) or not (`"no"`).

`race` race of respondent with levels: `"black"`, `"hispanic"` and `"other"`.

Among the 1500 observations, there are 1018 censoring times (`censoring = 0`) and 482 divorces (`censoring = 1`). Source: National Longitudinal Survey of Youth 1997 (NLSY97).

Usage

```
data(marriage_dataset)
```

Format

Time-to-event data.

References

Bureau of Labor Statistics, U.S. Department of Labor. National Longitudinal Survey of Youth 1997 cohort, 1997-2022 (rounds 1-20). Produced and distributed by the Center for Human Resource Research (CHRR), The Ohio State University. Columbus, OH: 2023.

plot.bayesCureModel *Plot method*

Description

Plots and computes HDIs.

Usage

```
## S3 method for class 'bayesCureModel'
plot(x, burn = NULL, alpha0 = 0.05, gamma_mix = TRUE,
 K_gamma = 5, plot_graphs = TRUE, bw = "nrd0", what = NULL, predict_output = NULL,
 index_of_main_mode = NULL, draw_legend = TRUE,...)
```

Arguments

x	An object of class bayesCureModel
burn	Number of iterations to discard as burn-in period.
alpha0	A value between 0 and 1 in order to compute the 1-$\alpha 0$ Highest Posterior Density regions.
gamma_mix	Boolean. If TRUE, the density of the marginal posterior distribution of the γ parameter is estimated from the sampled MCMC values by fitting a normal mixture model.
K_gamma	Used only when gamma_mix = TRUE and corresponds to the number of normal mixture components used to estimate the marginal posterior density of the γ parameter.
plot_graphs	Boolean, if FALSE only HDIs are computed.
bw	bandwidth
what	Integer or a character string with possible values equal to 'cured_prob', 'survival' or 'residuals'. An integer entry indicates which parameter should be plotted. If set to NULL (default), all parameters are plotted one by one. If set to 'cured_prob' or 'survival' the estimated cured probability or survival function is plotted, conditional on a set of covariates defined in the p_cured_output argument. In case where what = 'residuals' the residuals of the fitted model are plotted versus the quantity $-log(S)$ where S denotes the estimated survival function arising from the Kaplan-Meier estimate based on the residuals and the censoring times.
predict_output	Optional argument which is required only when what = 'cured_prob' or what = 'survival'. It is returned by a call to the predict.bayesCureModel function.
index_of_main_mode	If NULL (default), the whole MCMC output is used for plotting. Otherwise, it is a subset of the retained MCMC iterations in order to identify the main mode of the posterior distribution, as returned by the index_of_main_mode value of the summary.bayesCureRateModel function.

`draw_legend`	Boolean. If TRUE (default), a legend is plotted in the case where `what = 'survival'` or `what = 'cured_prob'`.
`...`	arguments passed by other methods.

Value

The function plots graphic output on the plot device if `plot_graphs = TRUE`. Furthermore, a list of $100(1-\alpha)\%$ Highest Density Intervals per parameter is returned.

Author(s)

Panagiotis Papastamoulis

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2, data = my_data_frame,
  promotion_time = list(family = 'exponential'),
  nChains = 2,
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
 mySummary <- summary(fit1, burn = 0)
 # plot the marginal posterior distribution of the first parameter in returned mcmc output
 plot(fit1, what = 1, burn = 0)
```

`plot.predict_bayesCureModel` *Plot method*

Description

Plot the output of the predict method.

Usage

```
## S3 method for class 'predict_bayesCureModel'
plot(x, what = 'survival', draw_legend = TRUE,...)
```

Arguments

x	An object of class predict_bayesCureModel
what	Character with possible values: 'cured_prob' or 'survival', corresponding to the estimated cured probability or survival function.
draw_legend	Boolean. If TRUE (default), a legend is plotted in the case where what = 'survival' or what = 'cured_prob'.
...	arguments passed by other methods.

Value

No value, just a plot.

Author(s)

Panagiotis Papastamoulis

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2, data = my_data_frame,
  promotion_time = list(family = 'exponential'),
  nChains = 2,
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
 #compute predictions for two individuals with
 # x1 = 0.2 and x2 = -1
 # and
 # x1 = -1 and x2 = 0
 covariate_levels1 <- data.frame(x1 = c(0.2,-1), x2 = c(-1,0))
 predictions <- predict(fit1, newdata = covariate_levels1, burn = 0)
 # plot cured probabilities based on the previous output
 plot(predictions, what='cured_prob')
```

predict.bayesCureModel *Predict method.*

Description

Returns MAP estimates of the survival function and the conditional cured probability for a given set of covariates.

Usage

```
## S3 method for class 'bayesCureModel'
predict(object, newdata = NULL, tau_values = NULL,
 burn = NULL, K_max = 1, alpha0 = 0.1, nDigits = 3, verbose = FALSE, ...)
```

Arguments

object	An object of class bayesCureModel
newdata	A data.frame with new data for the covariates. The column names as well as the class of each column (variable) should match with the input data.
tau_values	A vector of values for the response variable (time) for returning predictions for each row in the newdata.
burn	Positive integer corresponding to the number of mcmc iterations to discard as burn-in period
K_max	Maximum number of components in order to cluster the (univariate) values of the joint posterior distribution across the MCMC run. Used to identify the main mode of the posterior distribution.
alpha0	Scalar between 0 and 1 corresponding to 1 - confidencel level for computing Highest Density Intervals. If set to NULL, the confidence intervals are not computed.
nDigits	A positive integer for printing the output, after rounding to the corresponding number of digits. Default: nDigits = 3.
verbose	Boolean. If set to TRUE (default) the function prints a summary of the predictions.
...	ignored.

Details

The values of the posterior draws are clustered according to a (univariate) normal mixture model, and the main mode corresponds to the cluster with the largest mean. The maximum number of mixture components corresponds to the K_max argument. The **mclust** library is used for this purpose. The inference for the latent cure status of each (censored) observation is based on the MCMC draws corresponding to the main mode of the posterior distribution. The FDR is controlled according to the technique proposed in Papastamoulis and Rattray (2018).

In case where covariate_levels is set to TRUE, the summary function also returns a list named p_cured_output with the following entries

mcmc It is returned only in the case where the argument `covariate_values` is not `NULL`. A vector of posterior cured probabilities for the given values in `covariate_values`, per retained MCMC draw.

map It is returned only in the case where the argument `covariate_values` is not `NULL`. The cured probabilities computed at the MAP estimate of the parameters, for the given values `covariate_values`.

tau_values tau values

covariate_levels covariate levels

index_of_main_mode the subset of MCMC draws allocated to the main mode of the posterior distribution.

Value

A list with the following entries

`map_estimate` Maximum A Posteriori (MAP) estimate of the parameters of the model.

`highest_density_intervals` Highest Density Interval per parameter

`latent_cured_status` Estimated posterior probabilities of the latent cure status per censored subject.

`cured_at_given_FDR` Classification as cured or not, at given FDR level.

`p_cured_output` It is returned only in the case where the argument `covariate_values` is not `NULL`. See details.

`main_mode_index` The retained MCMC iterations which correspond to the main mode of the posterior distribution.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

Papastamoulis and Rattray (2018). A Bayesian Model Selection Approach for Identifying Differentially Expressed Transcripts from RNA Sequencing Data, Journal of the Royal Statistical Society Series C: Applied Statistics, Volume 67, Issue 1.

Scrucca L, Fraley C, Murphy TB, Raftery AE (2023). Model-Based Clustering, Classification, and Density Estimation Using mclust in R. Chapman and Hall/CRC. ISBN 978-1032234953

See Also

`cure_rate_MC3`

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2, data = my_data_frame,
  promotion_time = list(family = 'exponential'),
  nChains = 2,
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
 newdata <- data.frame(x1 = c(0.2,-1), x2 = c(-1,0))
 # return predicted values at tau = c(0.5, 1)
 my_prediction <- predict(fit1, newdata = newdata,
  burn = 0, tau_values = c(0.5, 1))
```

---

print.bayesCureModel *Print method*

Description

This function prints a summary of objects returned by the `cure_rate_MC3` function.

Usage

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

Arguments

`x`	An object of class `bayesCureModel`, which is returned by the `cure_rate_MC3` function.
`...`	ignored.

Details

The function prints some basic information for a `cure_rate_MC3`, such as the MAP estimate of model parameters and the value of Bayesian information criterion.

Value

No return value, called for side effects.

Author(s)

Panagiotis Papastamoulis

`print.predict_bayesCureModel` *Print method for the* `predict` *object*

Description

This function prints a summary of objects returned by the `predict.cure_rate_MC3` method.

Usage

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

Arguments

`x`	An object of class `predict_bayesCureModel`, which is returned by the `predict.cure_rate_MC3` method.
`...`	ignored.

Details

The function prints some basic information for the `predict` method of a `bayesCureModel` object.

Value

No return value, called for side effects.

Author(s)

Panagiotis Papastamoulis

---

print.summary_bayesCureModel
Print method for the summary

Description

This function prints a summary of objects returned by the `summary.cure_rate_MC3` method.

Usage

```
## S3 method for class 'summary_bayesCureModel'
print(x, digits = 2, ...)
```

Arguments

`x`	An object of class `summary_bayesCureModel`, which is returned by the `summary.cure_rate_MC3` method.
`digits`	Number of digits to print.
`...`	ignored.

Details

The function prints some basic information for the summary of a `bayesCureModel` object.

Value

No return value, called for side effects.

Author(s)

Panagiotis Papastamoulis

---

residuals.bayesCureModel
Computation of residuals.

Description

Methods for computing residuals for an object of class `bayesCureModel`. The Cox-Snell residuals are available for now.

Usage

```
## S3 method for class 'bayesCureModel'
residuals(object, type = "cox-snell",...)
```

Arguments

object	An object of class bayesCureModel
type	The type of residuals to be computed.
...	ignored.

Value

A vector of residuals.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

See Also

cure_rate_MC3

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2,
  data = my_data_frame,
  promotion_time = list(family = 'exponential'),
  nChains = 2,
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
 my_residuals <- residuals(fit1)
```

`sim_mix_data` *Simulated dataset*

Description

A synthetic dataset generated from a bimodal promotion time distribution. The available covariates are:

`x1` continuous.

`x2` factor with three levels.

Among the 500 observations, there are 123 censoring times (`censoring = 0`) and 377 "events" (`censoring = 1`). The true status (cured or susceptible) is contained in the column `true_status` and contains 59 cured and 441 susceptible subjects.

Usage

```
data(sim_mix_data)
```

Format

Time-to-event data.

`summary.bayesCureModel` *Summary method.*

Description

This function produces all summaries after fitting a cure rate model.

Usage

```
## S3 method for class 'bayesCureModel'
summary(object, burn = NULL, gamma_mix = TRUE,
 K_gamma = 3,  K_max = 3, fdr = 0.1, covariate_levels = NULL,
 yRange = NULL, alpha0 = 0.1, quantiles = c(0.05, 0.5, 0.95),
 verbose = FALSE, ...)
```

Arguments

`object`	An object of class `bayesCureModel`
`burn`	Positive integer corresponding to the number of mcmc iterations to discard as burn-in period
`gamma_mix`	Boolean. If TRUE, the density of the marginal posterior distribution of the γ parameter is estimated from the sampled MCMC values by fitting a normal mixture model.
`K_gamma`	Used only when `gamma_mix = TRUE` and corresponds to the number of normal mixture components used to estimate the marginal posterior density of the γ parameter.
`K_max`	Maximum number of components in order to cluster the (univariate) values of the joint posterior distribution across the MCMC run. Used to identify the main mode of the posterior distribution. See details.
`fdr`	The target value for controlling the False Discovery Rate when classifying subjects as cured or not.
`covariate_levels`	Optional `data.frame` with new data for the covariates. It is only required when the user wishes to obtain a vector with the estimated posterior cured probabilities for a given combination of covariates. The column names should be exactly the same with the ones used in the input data.
`yRange`	Optional range (a vector of two non-negative values) for computing the sequence of posterior probabilities for the given values in `covariate_levels`.
`alpha0`	Scalar between 0 and 1 corresponding to 1 - confidencel level for computing Highest Density Intervals. If set to NULL, the confidence intervals are not computed.
`quantiles`	A vector of quantiles to evaluate for each variable.
`verbose`	Boolean: if TRUE the function prints the summary.
`...`	ignored.

Details

The values of the posterior draws are clustered according to a (univariate) normal mixture model, and the main mode corresponds to the cluster with the largest mean. The maximum number of mixture components corresponds to the `K_max` argument. The **mclust** library is used for this purpose. The inference for the latent cure status of each (censored) observation is based on the MCMC draws corresponding to the main mode of the posterior distribution. The FDR is controlled according to the technique proposed in Papastamoulis and Rattray (2018).

In case where `covariate_levels` is set to `TRUE`, the `summary` function also returns a list named `p_cured_output` with the following entries

mcmc It is returned only in the case where the argument `covariate_values` is not `NULL`. A vector of posterior cured probabilities for the given values in `covariate_values`, per retained MCMC draw.

map It is returned only in the case where the argument `covariate_values` is not `NULL`. The cured probabilities computed at the MAP estimate of the parameters, for the given values `covariate_values`.

tau_values tau values

covariate_levels covariate levels

index_of_main_mode the subset of MCMC draws allocated to the main mode of the posterior distribution.

Value

A list with the following entries

`map_estimate` Maximum A Posteriori (MAP) estimate of the parameters of the model.

`highest_density_intervals` Highest Density Interval per parameter

`latent_cured_status` Estimated posterior probabilities of the latent cure status per censored subject.

`cured_at_given_FDR` Classification as cured or not, at given FDR level.

`p_cured_output` It is returned only in the case where the argument `covariate_values` is not `NULL`. See details.

`main_mode_index` The retained MCMC iterations which correspond to the main mode of the posterior distribution.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

Papastamoulis and Rattray (2018). A Bayesian Model Selection Approach for Identifying Differentially Expressed Transcripts from RNA Sequencing Data, Journal of the Royal Statistical Society Series C: Applied Statistics, Volume 67, Issue 1.

Scrucca L, Fraley C, Murphy TB, Raftery AE (2023). Model-Based Clustering, Classification, and Density Estimation Using mclust in R. Chapman and Hall/CRC. ISBN 978-1032234953

See Also

`cure_rate_MC3`

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
        n = 4
        # censoring indicators
        stat = rbinom(n, size = 1, prob = 0.5)
        # covariates
```

```
        x <- matrix(rnorm(2*n), n, 2)
        # observed response variable
        y <- rexp(n)
# define a data frame with the response and the covariates
        my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2,
  data = my_data_frame,
  promotion_time = list(family = 'exponential'),
  nChains = 2,
  nCores = 1,
  mcmc_cycles = 3, sweep=2)
 mySummary <- summary(fit1, burn = 0)
```

`summary.predict_bayesCureModel` *Summary method for predictions.*

Description

This function produces MCMC summaries for an object of class `predict_bayesCureModel`.

Usage

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

Arguments

`object`	An object of class `predict_bayesCureModel`.
`...`	Other options passed to the `summary.mcmc` method of the coda package.

Value

A list with the following entries

`survival`	MCMC summaries (quantiles) for the survival function.
`cured_probability`	MCMC summaries (quantiles) for the conditional cured probability.
`cumulative_hazard`	MCMC summaries (quantiles) for the cumulative hazard function.
`hazard_rate`	MCMC summaries (quantiles) for the hazard rate function.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis and Milienos (2024). Bayesian inference and cure rate modeling for event history data. TEST doi: 10.1007/s11749-024-00942-w.

See Also

cure_rate_MC3

Examples

```
# simulate toy data just for cran-check purposes
 set.seed(10)
 n = 4
 # censoring indicators
 stat = rbinom(n, size = 1, prob = 0.5)
 # covariates
 x <- matrix(rnorm(2*n), n, 2)
 # observed response variable
 y <- rexp(n)
#       define a data frame with the response and the covariates
 my_data_frame <- data.frame(y, stat, x1 = x[,1], x2 = x[,2])
# run a weibull model with default prior setup
# considering 2 heated chains
 fit1 <- cure_rate_MC3(survival::Surv(y, stat) ~ x1 + x2, data = my_data_frame,
      promotion_time = list(family = 'exponential'),
      nChains = 2,
      nCores = 1,
      mcmc_cycles = 3, sweep=2)
 newdata <- data.frame(x1 = c(0.2,-1), x2 = c(-1,0))
 # return predicted values at tau = c(0.5, 1)
 my_prediction <- predict(fit1, newdata = newdata,
      burn = 0, tau_values = c(0.5, 1))
 my_summary <- summary(my_prediction, quantiles = c(0.1,0.9))
```

Surv *Create a Survival Object*

Description

Create a survival object for use in survival analysis, as imported from the **survival** package.

Usage

```
Surv(time, time2, event, type = c("right", "left", "interval",
 "counting", "interval2", "mstate"), origin = 0)
```

Arguments

`time`	The follow-up time. For counting process data, this is the start time.
`time2`	The end time for counting process or interval-censored data. Optional for right-censored data.
`event`	The event indicator, normally 0=alive/censored, 1=dead/event occurred. Can also be a factor or logical.
`type`	Type of censoring. Options are `"right"`, `"left"`, `"interval"`, `"counting"`, `"interval2"`, or `"mstate"`.
`origin`	Starting point for time scale. Default is 0. Only used for type `"counting"`.

Details

The `Surv` function creates an object of class `"Surv"`, which is used to represent survival data. Depending on the arguments, the object can represent different types of censoring.

- Right-censoring: one time and event indicator.
- Left-censoring: similar to right-censoring but event=1 for censored.
- Interval-censoring: requires both `time` and `time2`.
- Counting process: both `time` and `time2` used to specify start and stop times.

The resulting object is used as a response in survival regression models and estimation functions.

Value

An object of class `"Surv"` which is used as a response in survival models.

Note

The implementation in the **bayesCureRateModel** package only supports right-censored data. The binary censoring indicators are interpreted as a time-to-event (1) or as a censoring time (0).

References

Therneau T (2024). A Package for Survival Analysis in R. R package version 3.7-0, https://CRAN.R-project.org/package=survival.

See Also

`cure_rate_MC3`

Examples

```
# Right-censored survival data
Surv(5, 1)
Surv(c(5, 10), c(1, 0))
```

Index