Package 'hdbayes'

November 15, 2025

Title Bayesian Analysis of Generalized Linear Models with Historical Data

Version 0.2.0

Description User-friendly functions for leveraging (multiple) historical data set(s) in Bayesian analysis of generalized linear models (GLMs) and survival models, along with support for Bayesian model averaging (BMA). The package provides functions for sampling from posterior distributions under various informative priors, including the prior induced by the Bayesian hierarchical model, power prior by Ibrahim and Chen (2000) <doi:10.1214/ss/1009212673>, normalized power prior by Duan et al. (2006) <doi:10.1002/env.752>, normalized asymptotic power prior by Ibrahim et al. (2015) <doi:10.1002/sim.6728>, commensurate prior by Hobbs et al. (2011) <doi:10.1111/j.1541-0420.2011.01564.x>, robust meta-analytic-predictive prior by Schmidli et al. (2014) <doi:10.1111/biom.12242>, latent exchangeability prior by Alt et al. (2024) <doi:10.1093/biomtc/ujae083>, and a normal (or half-normal) prior. The package also includes functions for computing model averaging weights, such as BMA, pseudo-BMA, pseudo-BMA with the Bayesian bootstrap, and stacking (Yao et al., 2018 <doi:10.1214/17-BA1091>), as well as for generating posterior samples from the ensemble distributions to reflect model uncertainty. In addition to GLMs, the package supports survival models including: (1) accelerated failure time (AFT) models, (2) piecewise exponential (PWE) models, i.e., proportional hazards models with piecewise constant baseline hazards, and (3) mixture cure rate models that assume a common probability of cure across subjects, paired with a PWE model for the non-cured population. Functions for computing marginal log-likelihoods under each implemented prior are also included. The package compiles all the 'CmdStan' models once during installation using the 'instantiate' package.

License MIT + file LICENSE

URL https://github.com/ethan-alt/hdbayes

BugReports https://github.com/ethan-alt/hdbayes/issues

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.2.0)

Imports instantiate (>= 0.1.0), callr, fs, formula.tools, stats, posterior, enrichwith, bridgesampling, mvtnorm, loo

Suggests cmdstanr (>= 0.6.0), ggplot2, ggthemes, knitr, parallel, rmarkdown, tibble, dplyr, survival

Additional_repositories https://mc-stan.org/r-packages/

SystemRequirements CmdStan (https://mc-stan.org/users/interfaces/cmdstan)

LazyData true

Collate 'E1684-data.R' 'E1690-data.R' 'E1694-data.R' 'E2696-data.R' 'IBCSG_curr-data.R' 'IBCSG_hist-data.R' 'actg019-data.R' 'actg036-data.R' 'data_checks_aft.R' 'get_stan_data_aft.R' 'aft_bhm.R' 'aft_loglik.R' 'aft_bhm_lognc.R' 'aft_commensurate.R' 'expfam_loglik.R' 'mixture_loglik.R' 'aft_commensurate_lognc.R' 'aft_leap.R' 'mixture_aft_loglik.R' 'aft_leap_lognc.R' 'aft_logml_commensurate.R' 'aft_logml_leap.R' 'aft_logml_map.R' 'aft_logml_npp.R' 'aft_logml_post.R' 'aft_pp_lognc.R' 'aft_logml_pp.R' 'aft_stratified_pp_lognc.R' 'aft_logml_stratified_pp.R' 'aft_npp_lognc.R' 'aft_npp.R' 'aft_post.R' 'aft_pp.R' 'aft_stratified_pp.R' 'compute_ensemble_weights.R' 'data_checks_pwe.R' 'get_stan_data_pwe.R' 'curepwe_bhm.R' 'pwe_loglik.R' 'curepwe_bhm_lognc.R' 'curepwe_commensurate.R' 'curepwe_commensurate_lognc.R' 'curepwe_leap.R' 'curepwe_leap_lognc.R' 'curepwe_logml_commensurate.R' 'curepwe_logml_leap.R' 'curepwe_logml_map.R' 'curepwe_logml_npp.R' 'curepwe_logml_post.R' 'curepwe_pp_lognc.R' 'curepwe_logml_pp.R' 'curepwe_stratified_pp_lognc.R' 'curepwe_logml_stratified_pp.R' 'curepwe_npp_lognc.R' 'curepwe_npp.R' 'curepwe_post.R' 'curepwe_pp.R' 'curepwe_stratified_pp.R' 'data_checks.R' 'get_stan_data.R' 'glm_bhm.R' 'glm_bhm_lognc.R' 'glm_commensurate.R' 'glm_commensurate_lognc.R' 'glm_leap.R' 'glm_leap_lognc.R' 'glm_logml_commensurate.R' 'glm_logml_leap.R' 'glm_logml_map.R' 'glm_logml_napp.R' 'glm_logml_npp.R' 'glm_logml_post.R' 'glm_pp_lognc.R' 'glm_logml_pp.R' 'glm_napp.R' 'glm_npp_lognc.R' 'glm_npp.R' 'glm_post.R' 'glm_pp.R' 'glm_rmap.R' 'hdbayes-package.R' 'lm_npp.R' 'pwe_bhm.R' 'pwe_bhm_lognc.R' 'pwe_commensurate.R' 'pwe_commensurate_lognc.R' 'pwe_leap.R' 'pwe_leap_lognc.R' 'pwe_logml_commensurate.R' 'pwe_logml_leap.R' 'pwe_logml_map.R'

'pwe_logml_npp.R' 'pwe_logml_post.R' 'pwe_pp_lognc.R'
'pwe_logml_pp.R' 'pwe_stratified_pp_lognc.R'
'pwe_logml_stratified_pp.R' 'pwe_npp_lognc.R' 'pwe_npp.R'
'pwe_post.R' 'pwe_pp.R' 'pwe_stratified_pp.R'
'sample_ensemble.R' 'zzz.R'

VignetteBuilder knitr

NeedsCompilation yes

Author Ethan M. Alt [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-6112-9030>),
Xinxin Chen [aut],
Luiz M. Carvalho [aut],
Joseph G. Ibrahim [aut],
Xiuya Chang [ctb]

Maintainer Ethan M. Alt <ethanalt@live.unc.edu>

Repository CRAN

Date/Publication 2025-11-15 12:10:02 UTC

Contents

actg019 *AIDS Clinical Trial ACTG019*

Description

A data set from the AIDS clinical trial ACTG019 (https://clinicaltrials.gov/ct2/show/NCT00000736) comparing zidovudine (AZT) with a placebo in adults with asymptomatic HIV infection. The study results were described in Volberding et al. (1990) doi:10.1056/NEJM199004053221401.

Usage

```
actg019
```

Format

A data frame with 822 rows and 5 variables:

outcome outcome variable with 1 indicating death, development of AIDS or AIDS-related complex (ARC) and 0 otherwise

age patient age in years

treatment treatment indicator, 0 = placebo, 1 = AZT

race race indicator, 0 = non-white, 1 = white

cd4 CD4 cell count

References

Volberding, P. A., Lagakos, S. W., Koch, M. A., Pettinelli, C., Myers, M. W., Booth, D. K., Balfour, H. H., Reichman, R. C., Bartlett, J. A., Hirsch, M. S., Murphy, R. L., Hardy, W. D., Soeiro, R., Fischl, M. A., Bartlett, J. G., Merigan, T. C., Hyslop, N. E., Richman, D. D., Valentine, F. T., Corey, L., and the AIDS Clinical Trials Group of the National Institute of Allergy and Infectious Diseases (1990). Zidovudine in asymptomatic human immunodeficiency virus infection. New England Journal of Medicine, 322(14), 941–949.

Chen, M.-H., Ibrahim, J. G., and Yiannoutsos, C. (1999). Prior elicitation, Variable Selection and Bayesian computation for Logistic Regression Models. Journal of the Royal Statistical Society Series B: Statistical Methodology, 61(1), 223–242.

actg036 *AIDS Clinical Trial ACTG036*

Description

A data set from the AIDS clinical trial ACTG036 (https://clinicaltrials.gov/study/NCT00001104) comparing zidovudine (AZT) with a placebo in patients with hereditary coagulation disorders and HIV infection. The study results were described in Merigan et al. (1991) doi:10.1182/blood.V78.4.900.900. This data set has the same variables as the actg019 data set. We can use the actg019 data as the historical data and the actg036 data as the current data.

Usage

```
actg036
```

Format

A data frame with 183 rows and 5 variables:

outcome outcome variable with 1 indicating death, development of AIDS or AIDS-related complex (ARC) and 0 otherwise

age patient age in years

treatment treatment indicator, 0 = placebo, 1 = AZT

race race indicator, 0 = non-white, 1 = white

cd4 CD4 cell count

References

Merigan, T., Amato, D., Balsley, J., Power, M., Price, W., Benoit, S., Perez-Michael, A., Brownstein, A., Kramer, A., and Brettler, D. (1991). Placebo-controlled trial to evaluate zidovudine in treatment of human immunodeficiency virus infection in asymptomatic patients with hemophilia. NHF-ACTG 036 Study Group. Blood, 78(4), 900–906.

Chen, M.-H., Ibrahim, J. G., and Yiannoutsos, C. (1999). Prior elicitation, Variable Selection and Bayesian computation for Logistic Regression Models. Journal of the Royal Statistical Society Series B: Statistical Methodology, 61(1), 223–242.

`aft.bhm` *Posterior of Bayesian hierarchical model (BHM)*

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using the Bayesian hierarchical model (BHM).

Usage

```
aft.bhm(
  formula,
  data.list,
  dist = "weibull",
  meta.mean.mean = NULL,
  meta.mean.sd = NULL,
  meta.sd.mean = NULL,
  meta.sd.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`data.list`	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting accelerated failure time (AFT) models, all historical data sets will be stacked into one historical data set.
`dist`	a character indicating the distribution of survival times. Currently, `dist` can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
`meta.mean.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, `meta.mean.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
`meta.mean.sd`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 10s.

meta.sd.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 0s.
meta.sd.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 1s.
scale.mean	location parameter for the half-normal prior on the scale parameters of current and historical data models. Defaults to 0.
scale.sd	scale parameter for the half-normal prior on the scale parameters of current and historical data models. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The Bayesian hierarchical model (BHM) assumes that the regression coefficients for the historical and current data are different, but are correlated through a common distribution, whose hyperparameters (i.e., mean and standard deviation (sd) (the covariance matrix is assumed to have a diagonal structure)) are treated as random. The number of regression coefficients for the current data is assumed to be the same as that for the historical data.

The hyperpriors on the mean and the sd hyperparameters are independent normal and independent half-normal distributions, respectively. The scale parameters for both current and historical data models are assumed to be independent and identically distributed, each assigned a half-normal prior.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
```

```
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    aft.bhm(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.commensurate` *Posterior of commensurate prior (CP)*

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using the commensurate prior (CP) by Hobbs et al. (2011) doi:10.1111/j.1541-0420.2011.01564.x.

Usage

```
aft.commensurate(
  formula,
  data.list,
  dist = "weibull",
  beta0.mean = NULL,
  beta0.sd = NULL,
  p.spike = 0.1,
  spike.mean = 200,
  spike.sd = 0.1,
  slab.mean = 0,
  slab.sd = 5,
  scale.mean = NULL,
  scale.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the survival::Surv(time, event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For fitting accelerated failure time (AFT) models, all historical data sets will be stacked into one historical data set.
dist	a character indicating the distribution of survival times. Currently, dist can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
beta0.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the prior on the historical data regression coefficients. If a scalar is provided, beta0.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta0.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the prior on the historical data regression coefficients. If a scalar is provided, same as for beta0.mean. Defaults to a vector of 10s.
p.spike	a scalar between 0 and 1 giving the probability of the spike component in spike-and-slab prior on commensurability parameter τ. Defaults to 0.1.
spike.mean	a scalar giving the location parameter for the half-normal prior (spike component) on τ. Defaults to 200.
spike.sd	a scalar giving the scale parameter for the half-normal prior (spike component) on τ. Defaults to 0.1.
slab.mean	a scalar giving the location parameter for the half-normal prior (slab component) on τ. Defaults to 0.
slab.sd	a scalar giving the scale parameter for the half-normal prior (slab component) on τ. Defaults to 5.
scale.mean	location parameter for the half-normal prior on the scale parameters of current and historical data models. Defaults to 0.
scale.sd	scale parameter for the half-normal prior on the scale parameters of current and historical data models. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The commensurate prior (CP) assumes that the regression coefficients for the current data model conditional on those for the historical data model are independent normal distributions with mean equal to the corresponding regression coefficients for the historical data and variance equal to the inverse of the corresponding elements of a vector of precision parameters (referred to as the commensurability parameter τ). We regard τ as random and elicit a spike-and-slab prior, which is specified as a mixture of two half-normal priors, on τ.

The number of current data regression coefficients is assumed to be the same as that of historical data regression coefficients. The scale parameters for both current and historical data models are assumed to be independent and identically distributed, each assigned a half-normal prior.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    aft.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      p.spike = 0.1,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.leap` *Posterior of latent exchangeability prior (LEAP)*

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using the latent exchangeability prior (LEAP) by Alt et al. (2024) doi:10.1093/biomtc/ujae083.

Usage

```
aft.leap(
  formula,
  data.list,
  dist = "weibull",
  K = 2,
  prob.conc = NULL,
  beta.mean = NULL,
  beta.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  gamma.lower = 0,
  gamma.upper = 1,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`data.list`	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting accelerated failure time (AFT) models, all historical data sets will be stacked into one historical data set.
`dist`	a character indicating the distribution of survival times. Currently, `dist` can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
`K`	the desired number of classes to identify. Defaults to 2.
`prob.conc`	a scalar or a vector of length `K` giving the concentration parameters for Dirichlet prior. If length == 2, a `Beta(prob.conc[1], prob.conc[2])` prior is used. If a scalar is provided, `prob.conc` will be a vector of repeated elements of the given scalar. Defaults to a vector of 1s.

`beta.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
`beta.sd`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
`scale.mean`	location parameter for the half-normal prior on the scale parameters for each class. Defaults to 0.
`scale.sd`	scale parameter for the half-normal prior on the scale parameters for each class. Defaults to 10.
`gamma.lower`	a scalar giving the lower bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 0.
`gamma.upper`	a scalar giving the upper bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 1.
`get.loglik`	whether to generate log-likelihood matrix. Defaults to FALSE.
`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The latent exchangeability prior (LEAP) discounts the historical data by identifying the most relevant individuals from the historical data. It is equivalent to a prior induced by the posterior of a finite mixture model for the historical data set.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    aft.leap(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      K = 2,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

---

aft.logml.commensurate

Log marginal likelihood of an accelerated failure time (AFT) model under the commensurate prior (CP)

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the commensurate prior (CP).

The arguments related to MCMC sampling are utilized to draw samples from the commensurate prior. These samples are then used to compute the logarithm of the normalizing constant of the commensurate prior using historical data sets.

Usage

```
aft.logml.commensurate(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from aft.commensurate() giving posterior samples of an AFT model under the commensurate prior (CP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "aft_commensurate"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the commensurate prior (CP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the CP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
```

```
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    d.cp = aft.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      p.spike = 0.1,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    aft.logml.commensurate(
      post.samples = d.cp,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.logml.leap` *Log marginal likelihood of an accelerated failure time (AFT) model under latent exchangeability prior (LEAP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the latent exchangeability prior (LEAP).

The arguments related to MCMC sampling are utilized to draw samples from the LEAP. These samples are then used to compute the logarithm of the normalizing constant of the LEAP using historical data sets.

Usage

```
aft.logml.leap(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from aft.leap() giving posterior samples of an AFT model under the latent exchangeability prior (LEAP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "aft_leap"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the latent exchangeability prior (LEAP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the LEAP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
```

```
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    d.leap = aft.leap(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      K= 2,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    aft.logml.leap(
      post.samples = d.leap,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.logml.map` *Log marginal likelihood of an accelerated failure time (AFT) model under the meta-analytic predictive (MAP) prior*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the meta-analytic predictive (MAP) prior. The MAP prior is equivalent to the prior induced by the Bayesian hierarchical model (BHM).

The arguments related to MCMC sampling are utilized to draw samples from the MAP prior. These samples are then used to compute the logarithm of the normalizing constant of the MAP prior using only historical data set.

Usage

```
aft.logml.map(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from aft.bhm() giving posterior samples of an AFT model under the Bayesian hierarchical model (BHM), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "aft_bhm"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the Bayesian hierarchical model (BHM) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the BHM using historical data set

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
```

```
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    d.bhm = aft.bhm(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    aft.logml.map(
      post.samples = d.bhm,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`aft.logml.npp` *Log marginal likelihood of an accelerated failure time (AFT) model under normalized power prior (NPP)*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the normalized power prior (NPP).

Usage

```
aft.logml.npp(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `aft.npp()` giving posterior samples of an AFT model under the normalized power prior (NPP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "aft_npp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the normalized power prior (NPP)

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if(requireNamespace("parallel")){
  library(parallel)
  ncores    = 2

  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin
  }

  a0 = seq(0, 1, length.out = 11)
  if (instantiate::stan_cmdstan_exists()) {
    ## call created function
    ## wrapper to obtain log normalizing constant in parallel package
    logncfun = function(a0, ...){
      hdbayes::aft.npp.lognc(
        formula = formula, histdata = data_list[[2]], a0 = a0, dist = "weibull",
        ...
      )
    }

    cl = makeCluster(ncores)
    clusterSetRNGStream(cl, 123)
    clusterExport(cl, varlist = c('formula', 'data_list'))
    a0.lognc = parLapply(
      cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
      iter_sampling = 1000, chains = 1, refresh = 0
    )
    stopCluster(cl)
    a0.lognc = data.frame( do.call(rbind, a0.lognc) )

    ## sample from normalized power prior
    d.npp = aft.npp(
```

```
    formula = formula,
    data.list = data_list,
    a0.lognc = a0.lognc$a0,
    lognc = a0.lognc$lognc,
    dist = "weibull",
    chains = 1, iter_warmup = 500, iter_sampling = 1000,
    refresh = 0
  )
  aft.logml.npp(
    post.samples = d.npp,
    bridge.args = list(silent = TRUE)
  )
 }
}
```

`aft.logml.post` *Log marginal likelihood of an accelerated failure time (AFT) model under a normal/half-normal prior*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the normal/half-normal prior.

Usage

```
aft.logml.post(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `aft.post()` giving posterior samples of an AFT model under the normal/half-normal prior, with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "aft_post"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the AFT model under the normal/half-normal prior

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1690)
    ## take subset for speed purposes
    E1690 = E1690[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690)
    d.post = aft.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      beta.sd = 10,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    aft.logml.post(
      post.samples = d.post,
      bridge.args = list(silent = TRUE)
    )
  }
}
```

---

aft.logml.pp *Log marginal likelihood of an accelerated failure time (AFT) model under the power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the PP using only historical data set.

Usage

```
aft.logml.pp(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
```

```
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from aft.pp() giving posterior samples of an AFT model under the power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

If the power prior parameter (a_0) is equal to zero, then the function will return the same result as the output from aft.logml.post().

If the power prior parameters (a_0) is non-zero, the function will return a list with the following objects

model "aft_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the power prior (PP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the PP using historical data set

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    d.pp = aft.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      a0 = 0.5,
      dist = "weibull",
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    aft.logml.pp(
      post.samples = d.pp,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`aft.logml.stratified.pp` *Log marginal likelihood of an accelerated failure time (AFT) model under the stratified power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of an AFT model under the stratified power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the stratified power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the stratified PP using only historical data sets.

Usage

```
aft.logml.stratified.pp(
  post.samples,
  bridge.args = NULL,
```

```
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from aft.stratified.pp() giving posterior samples of an AFT model under the stratified power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "aft_stratified_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the stratified power prior (PP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the stratified PP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y., & Yue, L. Q. (2019). Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. Journal of Biopharmaceutical Statistics, 29(5), 731–748.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    data_list = list(currdata = E1690, histdata = E1684)
    strata_list = list(rep(1:2, each = 25), rep(1:2, each = 50))
    # Alternatively, we can determine the strata based on propensity scores
    # using the psrwe package, which is available on GitHub
    d.stratified.pp = aft.stratified.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment,
      data.list = data_list,
      strata.list = strata_list,
      a0.strata = c(0.3, 0.5),
      dist = "weibull",
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    aft.logml.stratified.pp(
      post.samples = d.stratified.pp,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`aft.npp` *Posterior of normalized power prior (NPP)*

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using the normalized power prior (NPP) by Duan et al. (2006) doi:10.1002/env.752.

Usage

```
aft.npp(
  formula,
  data.list,
  a0.lognc,
  lognc,
  dist = "weibull",
  beta.mean = NULL,
```

```
  beta.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  a0.shape1 = 1,
  a0.shape2 = 1,
  a0.lower = 0,
  a0.upper = 1,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting accelerated failure time (AFT) models, all historical data sets will be stacked into one historical data set.
a0.lognc	a vector giving values of the power prior parameter for which the logarithm of the normalizing constant has been evaluated.
lognc	a vector giving the logarithm of the normalizing constant (as estimated by `aft.npp.lognc()` for each value of `a0.lognc` using the historical data set.
dist	a character indicating the distribution of survival times. Currently, `dist` can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
scale.mean	location parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 0.
scale.sd	scale parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 10.
a0.shape1	first shape parameter for the beta prior on the power prior parameter (a_0). When `a0.shape1 == 1` and `a0.shape2 == 1`, a uniform prior is used.
a0.shape2	second shape parameter for the beta prior on the power prior parameter (a_0). When `a0.shape1 == 1` and `a0.shape2 == 1`, a uniform prior is used.
a0.lower	a scalar giving the lower bound for a_0. Defaults to 0.

a0.upper	a scalar giving the upper bound for a_0. Defaults to 1.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

Before using this function, users must estimate the logarithm of the normalizing constant across a range of different values for the power prior parameter (a_0), possibly smoothing techniques over a find grid. The power prior parameters (a_0's) are treated as random with independent beta priors. The initial priors on the regression coefficients are independent normal priors. The current and historical data models are assumed to have a common scale parameter with a half-normal prior.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

See Also

aft.npp.lognc()

Examples

```
if(requireNamespace("parallel")){
  library(parallel)
  ncores    = 2

  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
```

```
  ## replace 0 failure times with 0.50 days
  E1684$failtime[E1684$failtime == 0] = 0.50/365.25
  E1690$failtime[E1690$failtime == 0] = 0.50/365.25
  E1684$cage = as.numeric(scale(E1684$age))
  E1690$cage = as.numeric(scale(E1690$age))
  data_list = list(currdata = E1690, histdata = E1684)
  formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin
}

a0 = seq(0, 1, length.out = 11)
if (instantiate::stan_cmdstan_exists()) {
  ## call created function
  ## wrapper to obtain log normalizing constant in parallel package
  logncfun = function(a0, ...){
    hdbayes::aft.npp.lognc(
      formula = formula, histdata = data_list[[2]], a0 = a0, dist = "weibull",
      ...
    )
  }

  cl = makeCluster(ncores)
  clusterSetRNGStream(cl, 123)
  clusterExport(cl, varlist = c('formula', 'data_list'))
  a0.lognc = parLapply(
    cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
    iter_sampling = 1000, chains = 1, refresh = 0
  )
  stopCluster(cl)
  a0.lognc = data.frame( do.call(rbind, a0.lognc) )

  ## sample from normalized power prior
  aft.npp(
    formula = formula,
    data.list = data_list,
    a0.lognc = a0.lognc$a0,
    lognc = a0.lognc$lognc,
    dist = "weibull",
    chains = 1, iter_warmup = 500, iter_sampling = 1000,
    refresh = 0
  )
}
}
```

`aft.npp.lognc` *Estimate the logarithm of the normalizing constant for normalized power prior (NPP) for one data set*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the normalizing constant of an accelerated failure time (AFT) model under the NPP for a fixed value of the power prior parameter $a_0 \in (0, 1)$ for one data set. The initial priors are independent normal priors on the regression coefficients and a half-normal prior on the scale parameter.

Usage

```
aft.npp.lognc(
  formula,
  histdata,
  a0,
  dist = "weibull",
  beta.mean = NULL,
  beta.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`histdata`	a `data.frame` giving the historical data.
`a0`	a scalar between 0 and 1 giving the (fixed) power prior parameter for the historical data.
`dist`	a character indicating the distribution of survival times. Currently, `dist` can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
`beta.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
`beta.sd`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
`scale.mean`	location parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 0.
`scale.sd`	scale parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 10.

bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a vector giving the value of a0, the estimated logarithm of the normalizing constant, the minimum estimated bulk effective sample size of the MCMC sampling, and the maximum Rhat.

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    aft.npp.lognc(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      histdata = E1684,
      a0 = 0.5,
      dist = "weibull",
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.post` *Posterior of a normal/half-normal prior*

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using a normal/half-normal prior.

Usage

```
aft.post(
  formula,
  data.list,
  dist = "weibull",
  beta.mean = NULL,
  beta.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`data.list`	a list consisting of one `data.frame` giving the current data. If `data.list` has more than one `data.frame`, only the first element will be used as the current data.
`dist`	a character indicating the distribution of survival times. Currently, `dist` can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
`beta.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
`beta.sd`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
`scale.mean`	location parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 0.

scale.sd	scale parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The priors on the regression coefficients are independent normal distributions. When the normal priors are elicited with large variances, the prior is also referred to as the reference or vague prior. The scale parameter is assumed to be independent of the regression coefficients with a half-normal prior.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1690)
    ## take subset for speed purposes
    E1690 = E1690[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690)
    aft.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      beta.sd = 10,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.pp` *Posterior of power prior (PP) with fixed a_0*

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using the power prior (PP) by Ibrahim and Chen (2000) doi:10.1214/ss/1009212673.

Usage

```
aft.pp(
  formula,
  data.list,
  a0,
  dist = "weibull",
  beta.mean = NULL,
  beta.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`data.list`	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting accelerated failure time (AFT) models, all historical data sets will be stacked into one historical data set.
`a0`	a scalar between 0 and 1 giving the (fixed) power prior parameter for the historical data.
`dist`	a character indicating the distribution of survival times. Currently, `dist` can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
`beta.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
scale.mean	location parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 0.
scale.sd	scale parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The power prior parameters (a_0's) are treated as fixed. The initial priors on the regression coefficients are independent normal priors. The current and historical data models are assumed to have a common scale parameter with a half-normal prior.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
```

```
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    aft.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      a0 = 0.5,
      dist = "weibull",
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`aft.stratified.pp` *Posterior of stratified power prior (PP) with fixed* a_0

Description

Sample from the posterior distribution of an accelerated failure time (AFT) model using the power prior (PP) within predefined strata. If the strata and power prior parameters (a_0's) are determined based on propensity scores, this function can be used to sample from the posterior of an AFT model under the propensity score-integrated power prior (PSIPP) by Wang et al. (2019) doi:10.1080/10543406.2019.1657133.

Usage

```
aft.stratified.pp(
  formula,
  data.list,
  strata.list,
  a0.strata,
  dist = "weibull",
  beta.mean = NULL,
  beta.sd = NULL,
  scale.mean = NULL,
  scale.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time,`

	event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For fitting accelerated failure time (AFT) models, all historical data sets will be stacked into one historical data set.
strata.list	a list of vectors specifying the stratum ID for each observation in the corresponding data set in data.list. The first element in the list corresponds to the current data, and the rest correspond to the historical data sets. Each vector should have the same length as the number of rows in the respective data set in data.list, with values representing stratum labels as positive integers (e.g., 1, 2, 3, ...).
a0.strata	A scalar or a vector of fixed power prior parameters (a_0's) for each stratum, with values between 0 and 1. If a scalar is provided, it will be replicated for all strata. If a vector is provided, its length must match the total number of unique strata across all data sets. The first element of a0.strata corresponds to stratum 1, the second to stratum 2, and so on.
dist	a character indicating the distribution of survival times. Currently, dist can be one of the following values: "weibull", "lognormal", or "loglogistic". Defaults to "weibull".
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
scale.mean	location parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 0.
scale.sd	scale parameter for the half-normal prior on the scale parameter of the AFT model. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The power prior parameters (a_0's) are treated as fixed and must be provided for each stratum. Users must also specify the stratum ID for each observation. Within each stratum, the initial priors on the regression coefficients are independent normal priors, and the current and historical data models are assumed to have a common scale parameter with a half-normal prior.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y., & Yue, L. Q. (2019). Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. Journal of Biopharmaceutical Statistics, 29(5), 731–748.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    data_list = list(currdata = E1690, histdata = E1684)
    strata_list = list(rep(1:2, each = 25), rep(1:2, each = 50))
    # Alternatively, we can determine the strata based on propensity scores
    # using the psrwe package, which is available on GitHub
    aft.stratified.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment,
      data.list = data_list,
      strata.list = strata_list,
      a0.strata = c(0.3, 0.5),
      dist = "weibull",
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`compute.ensemble.weights`

Compute model averaging weights

Description

Compute model averaging weights for a set of Bayesian models using Bayesian model averaging (BMA), pseudo-BMA, pseudo-BMA+ (pseudo-BMA with the Bayesian bootstrap), or stacking. This function takes a list of model fit objects, each containing posterior samples from a generalized linear model (GLM) or survival model, and returns normalized weights that can be used for model comparison or combining posterior samples using functions like `sample.ensemble()`.

Usage

```
compute.ensemble.weights(
  fit.list,
  type = c("bma", "pseudobma", "pseudobma+", "stacking"),
  prior.prob = NULL,
  bridge.args = NULL,
  loo.args = NULL,
  loo.wts.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`fit.list`	a list of model fit objects returned by functions in the `hdbayes` package. Each fit contains posterior samples from a generalized linear model (GLM) (e.g., via `glm.pp()`), an accelerated failure time (AFT) model (e.g., via `aft.pp()`), a piecewise exponential (PWE) model (e.g., via `pwe.pp()`), or a mixture cure rate model with a PWE component for the non-cured population (e.g., via `curepwe.pp()`). Each fit also includes two attributes: `data`, a list of variables specified in the data block of the Stan program, and `model`, a character string indicating the model name. To compute pseudo-BMA, pseudo-BMA+, or stacking weights, the fitting function must be called with `get.loglik = TRUE`.
`type`	a character string specifying the ensemble method used to compute model weights. Options are "bma" (Bayesian model averaging (BMA)), "pseudobma" (pseudo-BMA without the Bayesian bootstrap), "pseudobma+" (pseudo-BMA with the Bayesian bootstrap), and "stacking".
`prior.prob`	a numeric vector of prior model probabilities, used only when `type = "bma"`. Must be non-negative and sum to 1. If set to NULL, a uniform prior is used (i.e., all models are equally likely). Defaults to NULL.
`bridge.args`	a `list` of optional arguments (excluding `samples`, `log_posterior`, `data`, `lb`, and `ub`) to be passed to `bridgesampling::bridge_sampler()`. These arguments are used when estimating the log marginal likelihood, which is required if `type = "bma"`.
`loo.args`	a `list` of optional arguments (excluding `x`) to be passed to `loo::loo()` when computing pseudo-BMA, pseudo-BMA+, or stacking weights.

loo.wts.args	a list of optional arguments (excluding x, method, and BB) to be passed to loo::loo_model_weights() when computing pseudo-BMA, pseudo-BMA+, or stacking weights.
iter_warmup	number of warmup iterations to run per chain. Used only when computing the log marginal likelihood (i.e., when type = "bma"). Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Used only when computing the log marginal likelihood (i.e., when type = "bma"). Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Used only when computing the log marginal likelihood (i.e., when type = "bma"). Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init). These are used only when computing the log marginal likelihood (i.e., when type = "bma").

Details

The input fit.list should be a list of outputs from model fitting functions in the hdbayes package, such as glm.pp() (for generalized linear models), aft.pp() (for accelerated failure time models), pwe.pp() (for piecewise exponential (PWE) models), or curepwe.pp() (for mixture cure rate models with a PWE component for the non-cured population). To compute pseudo-BMA, pseudo-BMA+, or stacking weights, each fit must include pointwise log-likelihood values. To ensure this, the fitting function must be called with get.loglik = TRUE.

The arguments related to Markov chain Monte Carlo (MCMC) sampling are utilized to compute the logarithm of the normalizing constant for BMA, if applicable.

Value

The function returns a list with the following objects

weights a numeric vector of normalized model weights corresponding to the models in fit.list. The names of the weights are made unique based on the model identifiers.

type a character string indicating the method used to compute the model weights (e.g., "bma", "pseudobma", "pseudobma+", or "stacking")

res.logml a list of log marginal likelihood estimation results, returned only when type = "bma"

loo.list a list of outputs from loo::loo(), returned only when type is "pseudobma", "pseudobma+", or "stacking"

References

Yao, Y., Vehtari, A., Simpson, D., and Gelman, A. (2018). Using stacking to average Bayesian predictive distributions. Bayesian Analysis, 13(3), 917–1007.

Vehtari, A., Gelman, A., and Gabry, J. (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. Statistics and Computing, 27(5), 1413–1432.

See Also

sample.ensemble()

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    fit.pwe.pp = pwe.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      a0 = 0.5,
      get.loglik = TRUE,
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
    fit.pwe.post = pwe.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      get.loglik = TRUE,
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
    fit.aft.post = aft.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      dist = "weibull",
      beta.sd = 10,
      get.loglik = TRUE,
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
    compute.ensemble.weights(
      fit.list = list(fit.pwe.post, fit.pwe.pp, fit.aft.post),
      type = "pseudobma+",
      loo.args = list(save_psis = FALSE),
      loo.wts.args = list(optim_method="BFGS")
    )
```

```
  }
}
```

---

curepwe.bhm *Posterior of Bayesian hierarchical model (BHM)*

Description

Sample from the posterior distribution of a mixture cure rate model (referred to as the **CurePWE model**) using the Bayesian hierarchical model (BHM). The CurePWE model assumes that a fraction π of the population is "cured", while the remaining $1 - \pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1 - \pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model.

Usage

```
curepwe.bhm(
  formula,
  data.list,
  breaks,
  meta.mean.mean = NULL,
  meta.mean.sd = NULL,
  meta.sd.mean = NULL,
  meta.sd.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.

`data.list` a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting CurePWE models, all historical data sets will be stacked into one historical data set.

`breaks` a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

`meta.mean.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, `meta.mean.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

`meta.mean.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 10s.

`meta.sd.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 0s.

`meta.sd.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 1s.

`base.hazard.mean`
a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `meta.mean.mean`. Defaults to 0.

`base.hazard.sd` a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `meta.mean.mean`. Defaults to 10.

`logit.pcured.mean`
mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.

`logit.pcured.sd`
sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

`get.loglik` whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup` number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling` number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains` number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...` arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The Bayesian hierarchical model (BHM) assumes that the regression coefficients in the PWE models for the historical and current data are different, but are correlated through a common distribution, whose hyperparameters (i.e., mean and standard deviation (sd) (the covariance matrix is assumed to have a diagonal structure)) are treated as random. The number of regression coefficients for the current data is assumed to be the same as that for the historical data.

The hyperpriors on the mean and the sd hyperparameters are independent normal and independent half-normal distributions, respectively. The baseline hazard parameters for both current and historical data models are assumed to be independent and identically distributed (i.i.d.), each assigned a half-normal prior. Similarly, the cure fractions for both models are treated as i.i.d., with a normal prior specified on the logit of the cure fraction.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.bhm(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
```

```
  }
}
```

`curepwe.commensurate` *Posterior of commensurate prior (CP)*

Description

Sample from the posterior distribution of a mixture cure rate model (referred to as the **CurePWE model**) using the commensurate prior (CP) by Hobbs et al. (2011) doi:10.1111/j.1541-0420.2011.01564.x. The CurePWE model assumes that a fraction π of the population is "cured", while the remaining $1 - \pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1 - \pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model.

Usage

```
curepwe.commensurate(
  formula,
  data.list,
  breaks,
  beta0.mean = NULL,
  beta0.sd = NULL,
  p.spike = 0.1,
  spike.mean = 200,
  spike.sd = 0.1,
  slab.mean = 0,
  slab.sd = 5,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by

the survival::Surv(time, event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.

data.list a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For fitting CurePWE models, all historical data sets will be stacked into one historical data set.

breaks a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

beta0.mean a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the prior on the historical data regression coefficients. If a scalar is provided, beta0.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

beta0.sd a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the prior on the historical data regression coefficients. If a scalar is provided, same as for beta0.mean. Defaults to a vector of 10s.

p.spike a scalar between 0 and 1 giving the probability of the spike component in spike-and-slab prior on commensurability parameter τ. Defaults to 0.1.

spike.mean a scalar giving the location parameter for the half-normal prior (spike component) on τ. Defaults to 200.

spike.sd a scalar giving the scale parameter for the half-normal prior (spike component) on τ. Defaults to 0.1.

slab.mean a scalar giving the location parameter for the half-normal prior (slab component) on τ. Defaults to 0.

slab.sd a scalar giving the scale parameter for the half-normal prior (slab component) on τ. Defaults to 5.

base.hazard.mean
a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta0.mean. Defaults to 0.

base.hazard.sd a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta0.mean. Defaults to 10.

logit.pcured.mean
mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.

logit.pcured.sd
sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

get.loglik whether to generate log-likelihood matrix. Defaults to FALSE.

iter_warmup number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.

iter_sampling number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.

`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The commensurate prior (CP) assumes that the regression coefficients for the current data model conditional on those for the historical data model are independent normal distributions with mean equal to the corresponding regression coefficients for the historical data and variance equal to the inverse of the corresponding elements of a vector of precision parameters (referred to as the commensurability parameter τ). We regard τ as random and elicit a spike-and-slab prior, which is specified as a mixture of two half-normal priors, on τ.

The number of current data regression coefficients is assumed to be the same as that of historical data regression coefficients. The baseline hazard parameters for both current and historical data models are assumed to be independent and identically distributed (i.i.d.), each assigned a half-normal prior. Similarly, the cure fractions for both models are treated as i.i.d., with a normal prior specified on the logit of the cure fraction.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
```

```
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      p.spike = 0.1,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`curepwe.leap` *Posterior of latent exchangeability prior (LEAP)*

Description

Sample from the posterior distribution of a mixture cure rate model (referred to as the **CurePWE model**) using the latent exchangeability prior (LEAP) by Alt et al. (2024) doi:10.1093/biomtc/ujae083. The CurePWE model assumes that a fraction π of the population is "cured", while the remaining $1-\pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1-\pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model.

Usage

```
curepwe.leap(
  formula,
  data.list,
  breaks,
  K = 2,
  prob.conc = NULL,
  gamma.lower = 0,
  gamma.upper = 1,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
```

```
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the survival::Surv(time, event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For fitting CurePWE models, all historical data sets will be stacked into one historical data set.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
K	the desired number of classes to identify. Defaults to 2.
prob.conc	a scalar or a vector of length K giving the concentration parameters for Dirichlet prior. If length == 2, a Beta(prob.conc[1], prob.conc[2]) prior is used. If a scalar is provided, prob.conc will be a vector of repeated elements of the given scalar. Defaults to a vector of 1s.
gamma.lower	a scalar giving the lower bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 0.
gamma.upper	a scalar giving the upper bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 1.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
base.hazard.mean	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 0.
base.hazard.sd	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 10.
logit.pcured.mean	mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.

`logit.pcured.sd`
: sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

`get.loglik`
: whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup`
: number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling`
: number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains`
: number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...`
: arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The latent exchangeability prior (LEAP) discounts the historical data by identifying the most relevant individuals from the historical data. It is equivalent to a prior induced by the posterior of a finite mixture model for the historical data set.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
```

```
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.leap(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      K = 2,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`curepwe.logml.commensurate` *Log marginal likelihood of a mixture cure rate (CurePWE) under the commensurate prior (CP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the commensurate prior (CP).

The arguments related to MCMC sampling are utilized to draw samples from the commensurate prior. These samples are then used to compute the logarithm of the normalizing constant of the commensurate prior using historical data sets.

Usage

```
curepwe.logml.commensurate(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `curepwe.commensurate()` giving posterior samples of a CurePWE model under the commensurate prior (CP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Value

The function returns a `list` with the following objects

model "curepwe_commensurate"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the commensurate prior (CP) using all data sets

bs.hist an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the CP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
```

```
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.cp = curepwe.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      p.spike = 0.1,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    curepwe.logml.commensurate(
      post.samples = d.cp,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`curepwe.logml.leap` *Log marginal likelihood of a mixture cure rate (CurePWE) model under latent exchangeability prior (LEAP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the latent exchangeability prior (LEAP).

The arguments related to MCMC sampling are utilized to draw samples from the LEAP. These samples are then used to compute the logarithm of the normalizing constant of the LEAP using historical data sets.

Usage

```
curepwe.logml.leap(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `curepwe.leap()` giving posterior samples of a CurePWE model under the latent exchangeability prior (LEAP), with an attribute called 'data'

	which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "curepwe_leap"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the latent exchangeability prior (LEAP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the LEAP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
```

```
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.leap = curepwe.leap(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      K = 2,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    curepwe.logml.leap(
      post.samples = d.leap,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`curepwe.logml.map` *Log marginal likelihood of a mixture cure rate (CurePWE) model under the meta-analytic predictive (MAP) prior*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the meta-analytic predictive (MAP) prior. The MAP prior is equivalent to the prior induced by the Bayesian hierarchical model (BHM).

The arguments related to MCMC sampling are utilized to draw samples from the MAP prior. These samples are then used to compute the logarithm of the normalizing constant of the MAP prior using only historical data set.

Usage

```
curepwe.logml.map(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from curepwe.bhm() giving posterior samples of a CurePWE model under the Bayesian hierarchical model (BHM), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "curepwe_bhm"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the Bayesian hierarchical model (BHM) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the BHM using historical data set

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
```

```
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.bhm = curepwe.bhm(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    curepwe.logml.map(
      post.samples = d.bhm,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`curepwe.logml.npp` *Log marginal likelihood of a mixture cure rate (CurePWE) model under normalized power prior (NPP)*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the normalized power prior (NPP).

Usage

```
curepwe.logml.npp(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `curepwe.npp()` giving posterior samples of a CurePWE model under the normalized power prior (NPP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "curepwe_npp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the normalized power prior (NPP)

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if(requireNamespace("parallel")){
  library(parallel)
  ncores    = 2

  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin
  }

  a0 = seq(0, 1, length.out = 11)
  if (instantiate::stan_cmdstan_exists()) {
    ## call created function
    ## wrapper to obtain log normalizing constant in parallel package
    logncfun = function(a0, ...){
      hdbayes::curepwe.npp.lognc(
```

```
      formula = formula, histdata = data_list[[2]], breaks = breaks, a0 = a0,
      ...
    )
  }

  cl = makeCluster(ncores)
  clusterSetRNGStream(cl, 123)
  clusterExport(cl, varlist = c('formula', 'data_list', 'breaks'))
  a0.lognc = parLapply(
    cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
    iter_sampling = 1000, chains = 1, refresh = 0
  )
  stopCluster(cl)
  a0.lognc = data.frame( do.call(rbind, a0.lognc) )

  ## sample from normalized power prior
  d.npp = curepwe.npp(
    formula = formula,
    data.list = data_list,
    a0.lognc = a0.lognc$a0,
    lognc = a0.lognc$lognc,
    breaks = breaks,
    chains = 1, iter_warmup = 500, iter_sampling = 1000,
    refresh = 0
  )
  curepwe.logml.npp(
    post.samples = d.npp,
    bridge.args = list(silent = TRUE)
  )
 }
}
```

`curepwe.logml.post` *Log marginal likelihood of a mixture cure rate (CurePWE) model under a normal/half-normal prior*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the normal/half-normal prior.

Usage

```
curepwe.logml.post(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `curepwe.post()` giving posterior samples of a CurePWE model under the normal/half-normal prior, with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.

`bridge.args` a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "curepwe_post"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the CurePWE model under the normal/half-normal prior

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1690)
    ## take subset for speed purposes
    E1690 = E1690[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.post = curepwe.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    curepwe.logml.post(
      post.samples = d.post,
      bridge.args = list(silent = TRUE)
    )
  }
}
```

curepwe.logml.pp *Log marginal likelihood of a standard cure rate (CurePWE) model under the power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the PP using only historical data set.

Usage

```
curepwe.logml.pp(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from curepwe.pp() giving posterior samples of a CurePWE model under the power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

If the power prior parameter (a_0) is equal to zero, then the function will return the same result as the output from curepwe.logml.post().

If the power prior parameters (a_0) is non-zero, the function will return a list with the following objects

model "curepwe_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the power prior (PP) using all data sets

bs.hist an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the PP using historical data set

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.pp = curepwe.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      a0 = 0.5,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    curepwe.logml.pp(
      post.samples = d.pp,
      bridge.args = list(silent = TRUE),
```

```
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`curepwe.logml.stratified.pp` *Log marginal likelihood of a mixture cure rate (CurePWE) model under the stratified power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a CurePWE model under the stratified power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the stratified power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the stratified PP using only historical data sets.

Usage

```
curepwe.logml.stratified.pp(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `curepwe.stratified.pp()` giving posterior samples of a CurePWE model under the stratified power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.
`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Value

The function returns a `list` with the following objects

model "curepwe_stratified_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the stratified power prior (PP) using all data sets

bs.hist an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the stratified PP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y., & Yue, L. Q. (2019). Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. Journal of Biopharmaceutical Statistics, 29(5), 731–748.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    data_list = list(currdata = E1690, histdata = E1684)
    strata_list = list(rep(1:2, each = 25), rep(1:2, each = 50))
    # Alternatively, we can determine the strata based on propensity scores
    # using the psrwe package, which is available on GitHub
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.stratified.pp = curepwe.stratified.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment,
      data.list = data_list,
```

```
    strata.list = strata_list,
    breaks = breaks,
    a0.strata = c(0.3, 0.5),
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  curepwe.logml.stratified.pp(
    post.samples = d.stratified.pp,
    bridge.args = list(silent = TRUE),
    chains = 1, iter_warmup = 1000, iter_sampling = 2000
  )
 }
}
```

`curepwe.npp` *Posterior of normalized power prior (NPP)*

Description

Sample from the posterior distribution of a mixture cure rate model (referred to as the **CurePWE model**) using the normalized power prior (NPP) by Duan et al. (2006) doi:10.1002/env.752. The CurePWE model assumes that a fraction π of the population is "cured", while the remaining $1 - \pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1 - \pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model.

Usage

```
curepwe.npp(
  formula,
  data.list,
  a0.lognc,
  lognc,
  breaks,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  a0.shape1 = 1,
  a0.shape2 = 1,
  a0.lower = 0,
  a0.upper = 1,
  get.loglik = FALSE,
```

```
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting CurePWE models, all historical data sets will be stacked into one historical data set.
a0.lognc	a vector giving values of the power prior parameter for which the logarithm of the normalizing constant has been evaluated.
lognc	a vector giving the logarithm of the normalizing constant (as estimated by `pwe.npp.lognc()` for each value of `a0.lognc` using the historical data set.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
base.hazard.mean	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.
base.hazard.sd	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.
logit.pcured.mean	mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.
logit.pcured.sd	sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.
a0.shape1	first shape parameter for the beta prior on the power prior parameter (a_0). When `a0.shape1 == 1` and `a0.shape2 == 1`, a uniform prior is used.

a0.shape2	second shape parameter for the beta prior on the power prior parameter (a_0). When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
a0.lower	a scalar giving the lower bound for a_0. Defaults to 0.
a0.upper	a scalar giving the upper bound for a_0. Defaults to 1.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

Before using this function, users must estimate the logarithm of the normalizing constant across a range of different values for the power prior parameter (a_0), possibly smoothing techniques over a find grid. The power prior parameters (a_0's) are treated as random with independent beta priors. The initial priors on the regression coefficients are independent normal priors. The current and historical data models are assumed to share the baseline hazard parameters with half-normal priors. Additionally, a normal prior is specified for the logit of the cure fraction π.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

See Also

curepwe.npp.lognc()

Examples

```
if(requireNamespace("parallel")){
  library(parallel)
  ncores    = 2

  if(requireNamespace("survival")){
    library(survival)
```

```
  data(E1684)
  data(E1690)
  ## take subset for speed purposes
  E1684 = E1684[1:100, ]
  E1690 = E1690[1:50, ]
  ## replace 0 failure times with 0.50 days
  E1684$failtime[E1684$failtime == 0] = 0.50/365.25
  E1690$failtime[E1690$failtime == 0] = 0.50/365.25
  E1684$cage = as.numeric(scale(E1684$age))
  E1690$cage = as.numeric(scale(E1690$age))
  data_list = list(currdata = E1690, histdata = E1684)
  nbreaks = 3
  probs   = 1:nbreaks / nbreaks
  breaks  = as.numeric(
    quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
  )
  breaks  = c(0, breaks)
  breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
  formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin
}

a0 = seq(0, 1, length.out = 11)
if (instantiate::stan_cmdstan_exists()) {
  ## call created function
  ## wrapper to obtain log normalizing constant in parallel package
  logncfun = function(a0, ...){
    hdbayes::curepwe.npp.lognc(
      formula = formula, histdata = data_list[[2]], breaks = breaks, a0 = a0,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      ...
    )
  }

  cl = makeCluster(ncores)
  clusterSetRNGStream(cl, 123)
  clusterExport(cl, varlist = c('formula', 'data_list', 'breaks'))
  a0.lognc = parLapply(
    cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
    iter_sampling = 1000, chains = 1, refresh = 0
  )
  stopCluster(cl)
  a0.lognc = data.frame( do.call(rbind, a0.lognc) )

  ## sample from normalized power prior
  curepwe.npp(
    formula = formula,
    data.list = data_list,
    a0.lognc = a0.lognc$a0,
    lognc = a0.lognc$lognc,
    breaks = breaks,
    logit.pcured.mean = 0, logit.pcured.sd = 3,
    chains = 1, iter_warmup = 500, iter_sampling = 1000,
    refresh = 0
```

```
    )
  }
}
```

`curepwe.npp.lognc` *Estimate the logarithm of the normalizing constant for normalized power prior (NPP) for one data set*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the normalizing constant of a mixture cure rate (CurePWE) model under the NPP for a fixed value of the power prior parameter $a_0 \in (0, 1)$ for one data set. The initial priors are independent normal priors on the regression coefficients and half-normal priors on the baseline hazard parameters. Additionally, a normal prior is specified for the logit of the cure fraction π.

Usage

```
curepwe.npp.lognc(
  formula,
  histdata,
  breaks,
  a0,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`histdata`	a `data.frame` giving the historical data.
`breaks`	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

`a0` a scalar between 0 and 1 giving the (fixed) power prior parameter for the historical data.

`beta.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

`beta.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.

`base.hazard.mean` a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.

`base.hazard.sd` a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.

`logit.pcured.mean` mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.

`logit.pcured.sd` sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

`bridge.args` a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

`iter_warmup` number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling` number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains` number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...` arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Value

The function returns a vector giving the value of a0, the estimated logarithm of the normalizing constant, the minimum estimated bulk effective sample size of the MCMC sampling, and the maximum Rhat.

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1684[E1684$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.npp.lognc(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      histdata = E1684,
      breaks = breaks,
      a0 = 0.5,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`curepwe.post` *Posterior of a normal/half-normal prior*

Description

Sample from the posterior distribution of a mixture cure rate model using a normal/half-normal prior. The model assumes that a fraction π of the population is "cured", while the remaining $1 - \pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1 - \pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model. This cure rate model is referred to as the **CurePWE model**.

Usage

```
curepwe.post(
  formula,
```

```
  data.list,
  breaks,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
`data.list`	a list consisting of one `data.frame` giving the current data. If `data.list` has more than one `data.frame`, only the first element will be used as the current data.
`breaks`	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
`beta.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
`beta.sd`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
`base.hazard.mean`	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.
`base.hazard.sd`	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.
`logit.pcured.mean`	mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.
`logit.pcured.sd`	sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The priors on the regression coefficients in the PWE model are independent normal distributions. When the normal priors are elicited with large variances, the prior is also referred to as the reference or vague prior. The baseline hazard parameters of the PWE model are assumed to be independent of the regression coefficients with half-normal priors. Additionally, a normal prior is specified for the logit of the cure fraction π.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1690)
    ## take subset for speed purposes
    E1690 = E1690[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
```

```
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

---

curepwe.pp *Posterior of power prior (PP) with fixed a_0*

Description

Sample from the posterior distribution of a mixture cure rate model (referred to as the **CurePWE model**) using the power prior (PP) by Ibrahim and Chen (2000) doi:10.1214/ss/1009212673. The CurePWE model assumes that a fraction π of the population is "cured", while the remaining $1-\pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1-\pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model.

Usage

```
curepwe.pp(
  formula,
  data.list,
  breaks,
  a0,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the survival::Surv(time, event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.

`data.list` a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting CurePWE models, all historical data sets will be stacked into one historical data set.

`breaks` a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

`a0` a scalar between 0 and 1 giving the (fixed) power prior parameter for the historical data.

`beta.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

`beta.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.

`base.hazard.mean`
a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.

`base.hazard.sd` a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.

`logit.pcured.mean`
mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.

`logit.pcured.sd`
sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

`get.loglik` whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup` number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling` number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains` number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...` arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The power prior parameters (a_0's) are treated as fixed. The initial priors on the regression coefficients are independent normal priors. The current and historical data models are assumed to share the baseline hazard parameters with half-normal priors. Additionally, a normal prior is specified for the logit of the cure fraction π.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      a0 = 0.5,
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`curepwe.stratified.pp` *Posterior of stratified power prior (PP) with fixed* a_0

Description

Sample from the posterior distribution of a mixture cure rate model (referred to as the **CurePWE model**) using the power prior (PP) within predefined strata. If the strata and power prior parameters (a_0's) are determined based on propensity scores, this function can be used to sample from the posterior of a CurePWE model under the propensity score-integrated power prior (PSIPP) by Wang et al. (2019) doi:10.1080/10543406.2019.1657133. The CurePWE model assumes that a fraction π of the population is "cured", while the remaining $1 - \pi$ are susceptible to the event of interest. The survival function for the entire population is given by:

$$S_{\text{pop}}(t) = \pi + (1 - \pi)S(t),$$

where $S(t)$ represents the survival function of the non-cured individuals. We model $S(t)$ using a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard). Covariates are incorporated through the PWE model.

Usage

```
curepwe.stratified.pp(
  formula,
  data.list,
  strata.list,
  breaks,
  a0.strata,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  logit.pcured.mean = NULL,
  logit.pcured.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates in the PWE model. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.

`data.list` a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting CurePWE models, all historical data sets will be stacked into one historical data set.

`strata.list` a list of vectors specifying the stratum ID for each observation in the corresponding data set in `data.list`. The first element in the list corresponds to the current data, and the rest correspond to the historical data sets. Each vector should have the same length as the number of rows in the respective data set in `data.list`, with values representing stratum labels as positive integers (e.g., 1, 2, 3, ...).

`breaks` a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

`a0.strata` A scalar or a vector of fixed power prior parameters (a_0's) for each stratum, with values between 0 and 1. If a scalar is provided, it will be replicated for all strata. If a vector is provided, its length must match the total number of unique strata across all data sets. The first element of `a0.strata` corresponds to stratum 1, the second to stratum 2, and so on.

`beta.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

`beta.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.

`base.hazard.mean`
a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.

`base.hazard.sd` a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.

`logit.pcured.mean`
mean parameter for the normal prior on the logit of the cure fraction π. Defaults to 0.

`logit.pcured.sd`
sd parameter for the normal prior on the logit of the cure fraction π. Defaults to 3.

`get.loglik` whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup` number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling` number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains` number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...` arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The power prior parameters (a_0's) are treated as fixed and must be provided for each stratum. Users must also specify the stratum ID for each observation. Within each stratum, the initial priors on the regression coefficients are independent normal priors, and the current and historical data models are assumed to share the baseline hazard parameters with half-normal priors. Additionally, a normal prior is specified for the logit of the cure fraction within each stratum.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y., & Yue, L. Q. (2019). Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. Journal of Biopharmaceutical Statistics, 29(5), 731–748.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    data_list = list(currdata = E1690, histdata = E1684)
    strata_list = list(rep(1:2, each = 25), rep(1:2, each = 50))
    # Alternatively, we can determine the strata based on propensity scores
    # using the psrwe package, which is available on GitHub
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    curepwe.stratified.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment,
      data.list = data_list,
      strata.list = strata_list,
      breaks = breaks,
      a0.strata = c(0.3, 0.5),
```

```
      logit.pcured.mean = 0, logit.pcured.sd = 3,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

E1684 *ECOG E1684 Trial*

Description

A data set from the ECOG E1684 trial comparing the high-dose interferon alfa-2b (IFN) therapy with the observation in resected high-risk melanoma patients. The study results were described in Kirkwood et al. (1996) doi:10.1200/JCO.1996.14.1.7.

Usage

```
E1684
```

Format

A data frame with 262 rows and 8 variables:

failtime time to relapse in years

failcens censoring indicator for time to relapse, 0 = did not relapse, 1 = relapsed

survtime time to death in years

survcens censoring indicator for time to death, 0 = alive, 1 = dead

treatment treatment indicator, 0 = observation, 1 = high-dose IFN

sex gender indicator, 0 = male, 1 = female

age patient age in years

node_bin indicator for having more than one cancerous lymph node, 0 = with one or no cancerous lymph nodes, 1 = with more than one cancerous lymph node

References

Kirkwood, J. M., Strawderman, M. H., Ernstoff, M. S., Smith, T. J., Borden, E. C., and Blum, R. H. (1996). Interferon alfa-2b adjuvant therapy of high-risk resected cutaneous melanoma: The Eastern Cooperative Oncology Group trial EST 1684. Journal of Clinical Oncology, 14(1), 7–17.

E1690 *ECOG E1690 Trial*

Description

A data set from the ECOG E1690 trial evaluating the effectiveness of the interferon alfa-2b (IFN) therapy compared to the observation in high-risk melanoma patients. There were three arms in the trial: high-dose IFN, low-dose IFN, and observation. The study results were described in Kirkwood et al. (2000) doi:10.1200/JCO.2000.18.12.2444. Here, we only consider the high-dose IFN arm and the observation arm so that this data set has the same variables as the E1684 data set. We can use the E1684 data as the historical data and the E1690 data as the current data.

Usage

```
E1690
```

Format

A data frame with 426 rows and 8 variables:

failtime time to relapse in years

failcens censoring indicator for time to relapse, 0 = did not relapse, 1 = relapsed

survtime time to death in years

survcens censoring indicator for time to death, 0 = alive, 1 = dead

treatment treatment indicator, 0 = observation, 1 = high-dose IFN

sex gender indicator, 0 = male, 1 = female

age patient age in years

node_bin indicator for having more than one cancerous lymph node, 0 = with one or no cancerous lymph nodes, 1 = with more than one cancerous lymph node

References

Kirkwood, J. M., Ibrahim, J. G., Sondak, V. K., Richards, J., Flaherty, L. E., Ernstoff, M. S., Smith, T. J., Rao, U., Steele, M., and Blum, R. H. (2000). High- and low-dose interferon alfa-2b in high-risk melanoma: First analysis of intergroup trial E1690/S9111/C9190. Journal of Clinical Oncology, 18(12), 2444–2458.

E1694 *ECOG E1694 Trial*

Description

A data set from the ECOG E1694 trial comparing the GM2-KLH/QS-21 (GMK) vaccine with high-dose interferon alfa-2b (IFN) therapy in resected high-risk melanoma patients. The study results were described in Kirkwood et al. (2001) doi:10.1200/JCO.2001.19.9.2370. This data set only includes patients without nodal metastasis and has the same variables as the E2696 data set. We can use the E2696 data as the historical data and the E1694 data as the current data.

Usage

```
E1694
```

Format

A data frame with 200 rows and 6 variables:

failtime relapse-free survival (RFS) times (in months)

failind relapse indicator, 0 = right censored, 1 = relapse

treatment treatment indicator, 0 = GMK, 1 = IFN

sex gender indicator, 0 = male, 1 = female

age patient age in years

perform ECOG performance status indicator, 0 = fully active patient, able to carry on all pre-disease performance without restriction, 1 = restricted in physically strenuous activity, but are ambulatory and able to carry out work of a light or sedentary nature

References

Kirkwood, J. M., Ibrahim, J. G., Sosman, J. A., Sondak, V. K., Agarwala, S. S., Ernstoff, M. S., and Rao, U. (2001). High-dose interferon alfa-2b significantly prolongs relapse-free and overall survival compared with the GM2-KLH/QS-21 vaccine in patients with resected stage IIB-III melanoma: Results of intergroup trial E1694/S9512/C509801. Journal of Clinical Oncology, 19(9), 2370–2380.

E2696 *ECOG E2696 Trial*

Description

A data set from the ECOG E2696 trial comparing the combination of the GM2-KLH/QS-21 (GMK) vaccine and high-dose interferon alfa-2b (IFN) therapy with the GMK vaccine alone in resected high-risk melanoma patients. The study results were described in Kirkwood et al. (2001) doi:10.1200/JCO.2001.19.5.1430. This data set only includes patients without nodal metastasis.

Usage

```
E2696
```

Format

A data frame with 105 rows and 6 variables:

failtime relapse-free survival (RFS) times (in months)

failind relapse indicator, 0 = right censored, 1 = relapse

treatment treatment indicator, 0 = GMK, 1 = GMK and IFN

sex gender indicator, 0 = male, 1 = female

age patient age in years

perform ECOG performance status indicator, 0 = fully active patient, able to carry on all pre-disease performance without restriction, 1 = restricted in physically strenuous activity, but are ambulatory and able to carry out work of a light or sedentary nature

References

Kirkwood, J. M., Ibrahim, J., Lawson, D. H., Atkins, M. B., Agarwala, S. S., Collins, K., Mascari, R., Morrissey, D. M., and Chapman, P. B. (2001). High-dose interferon alfa-2b does not diminish antibody response to GM2 vaccination in patients with resected melanoma: Results of the multicenter eastern cooperative oncology group phase II trial E2696. Journal of Clinical Oncology, 19(5), 1430–1436.

---

glm.bhm *Posterior of Bayesian hierarchical model (BHM)*

Description

Sample from the posterior distribution of a GLM using the Bayesian hierarchical model (BHM).

Usage

```
glm.bhm(
  formula,
  family,
  data.list,
  offset.list = NULL,
  meta.mean.mean = NULL,
  meta.mean.sd = NULL,
  meta.sd.mean = NULL,
  meta.sd.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
```

```
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets.
offset.list	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
meta.mean.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, meta.mean.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
meta.mean.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 10s.
meta.sd.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 0s.
meta.sd.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 1s.
disp.mean	a scalar or a vector whose dimension is equal to the number of data sets (including the current data) giving the location parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 0s.
disp.sd	a scalar or a vector whose dimension is equal to the number of data sets (including the current data) giving the scale parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 10s.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.

chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The Bayesian hierarchical model (BHM) assumes that the regression coefficients for the historical and current data are different, but are correlated through a common distribution, whose hyperparameters (i.e., mean and standard deviation (sd) (the covariance matrix is assumed to have a diagonal structure)) are treated as random. The number of regression coefficients for the current data is assumed to be the same as that for the historical data.

The hyperpriors on the mean and the sd hyperparameters are independent normal and independent half-normal distributions, respectively. The priors on the dispersion parameters (if applicable) for the current and historical data sets are independent half-normal distributions.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  data_list = list(currdata = actg019, histdata = actg036)
  glm.bhm(
    formula = outcome ~ scale(age) + race + treatment + scale(cd4),
    family = binomial('logit'),
    data.list = data_list,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

---

glm.commensurate *Posterior of commensurate prior (CP)*

Description

Sample from the posterior distribution of a GLM using the commensurate prior (CP) by Hobbs et al. (2011) doi:10.1111/j.1541-0420.2011.01564.x.

Usage

```
glm.commensurate(
  formula,
  family,
  data.list,
  offset.list = NULL,
  beta0.mean = NULL,
  beta0.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  p.spike = 0.1,
  spike.mean = 200,
  spike.sd = 0.1,
  slab.mean = 0,
  slab.sd = 5,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates
family	an object of class family. See ?stats::family
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets.
offset.list	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
beta0.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the prior on the historical data regression coefficients. If a scalar is provided, beta0.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta0.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the prior on the historical data regression coefficients. If a scalar is provided, same as for beta0.mean. Defaults to a vector of 10s.
disp.mean	a scalar or a vector whose dimension is equal to the number of data sets (including the current data) giving the location parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for beta0.mean. Defaults to a vector of 0s.

`disp.sd`	a scalar or a vector whose dimension is equal to the number of data sets (including the current data) giving the scale parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for `beta0.mean`. Defaults to a vector of 10s.
`p.spike`	a scalar between 0 and 1 giving the probability of the spike component in spike-and-slab prior on commensurability parameter τ. Defaults to 0.1.
`spike.mean`	a scalar giving the location parameter for the half-normal prior (spike component) on τ. Defaults to 200.
`spike.sd`	a scalar giving the scale parameter for the half-normal prior (spike component) on τ. Defaults to 0.1.
`slab.mean`	a scalar giving the location parameter for the half-normal prior (slab component) on τ. Defaults to 0.
`slab.sd`	a scalar giving the scale parameter for the half-normal prior (slab component) on τ. Defaults to 5.
`get.loglik`	whether to generate log-likelihood matrix. Defaults to FALSE.
`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The commensurate prior (CP) assumes that the regression coefficients for the current data conditional on those for the historical data are independent normal distributions with mean equal to the corresponding regression coefficients for the historical data and variance equal to the inverse of the corresponding elements of a vector of precision parameters (referred to as the commensurability parameter τ). We regard τ as random and elicit a spike-and-slab prior, which is specified as a mixture of two half-normal priors, on τ.

The number of current data regression coefficients is assumed to be the same as that of historical data regression coefficients. The priors on the dispersion parameters (if applicable) for the current and historical data sets are independent half-normal distributions.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  data_list = list(currdata = actg019, histdata = actg036)
  glm.commensurate(
    formula = cd4 ~ treatment + age + race,
    family = poisson(), data.list = data_list,
    p.spike = 0.1,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

`glm.leap` *Posterior of latent exchangeability prior (LEAP)*

Description

Sample from the posterior distribution of a GLM using the latent exchangeability prior (LEAP) by Alt et al. (2024) doi:10.1093/biomtc/ujae083.

Usage

```
glm.leap(
  formula,
  family,
  data.list,
  K = 2,
  prob.conc = NULL,
  offset.list = NULL,
  beta.mean = NULL,
  beta.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  gamma.lower = 0,
  gamma.upper = 1,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
```

```
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For LEAP implementation, all historical data sets will be stacked into one historical data set.
K	the desired number of classes to identify. Defaults to 2.
prob.conc	a scalar or a vector of length K giving the concentration parameters for Dirichlet prior. If length == 2, a Beta(prob.conc[1], prob.conc[2]) prior is used. If a scalar is provided, prob.conc will be a vector of repeated elements of the given scalar. Defaults to a vector of 1s.
offset.list	a list of matrices giving the offset for current data followed by historical data. For each matrix, the number of rows corresponds to observations and columns correspond to classes. Defaults to a list of matrices of 0s. Note that the first element of offset.list (corresponding to the offset for current data) should be a matrix of repeated columns if offset.list is not NULL.
beta.mean	a scalar or a p x K matrix of mean parameters for initial prior on regression coefficients, where p is the number of regression coefficients (including intercept). If a scalar is provided, beta.mean will be a matrix of repeated elements of the given scalar. Defaults to a matrix of 0s.
beta.sd	a scalar or a p x K matrix of sd parameters for the initial prior on regression coefficients, where p is the number of regression coefficients (including intercept). If a scalar is provided, same as for beta.mean. Defaults to a matrix of 10s.
disp.mean	a scalar or a vector whose dimension is equal to the number of classes (K) giving the location parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, disp.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
disp.sd	a scalar or a vector whose dimension is equal to the number of classes (K) giving the scale parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for disp.mean. Defaults to a vector of 10s.
gamma.lower	a scalar giving the lower bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 0.
gamma.upper	a scalar giving the upper bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 1.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.

chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The latent exchangeability prior (LEAP) discounts the historical data by identifying the most relevant individuals from the historical data. It is equivalent to a prior induced by the posterior of a finite mixture model for the historical data set.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Examples

```
data(actg019)
data(actg036)
# take subset for speed purposes
actg019 = actg019[1:100, ]
actg036 = actg036[1:50, ]
if (instantiate::stan_cmdstan_exists()) {
  glm.leap(
    formula = outcome ~ scale(age) + race + treatment + scale(cd4),
    family = binomial('logit'),
    data.list = list(actg019, actg036),
    K = 2,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

---

glm.logml.commensurate

Log marginal likelihood of a GLM under commensurate prior (CP)

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the commensurate prior (CP).

The arguments related to MCMC sampling are utilized to draw samples from the commensurate prior. These samples are then used to compute the logarithm of the normalizing constant of the commensurate prior using historical data sets.

Usage

```
glm.logml.commensurate(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `glm.commensurate()` giving posterior samples of a GLM under the commensurate prior (CP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.
`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Value

The function returns a `list` with the following objects

model "glm_commensurate"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the commensurate prior (CP) using all data sets

bs.hist an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the CP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019   = actg019[1:100, ]
  actg036   = actg036[1:50, ]
  formula   = cd4 ~ treatment + age + race
  family    = poisson()
  data_list = list(currdata = actg019, histdata = actg036)
  d.cp      = glm.commensurate(
    formula = formula,
    family = family,
    data.list = data_list,
    p.spike = 0.1,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  glm.logml.commensurate(
    post.samples = d.cp,
    bridge.args = list(silent = TRUE),
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

---

glm.logml.leap *Log marginal likelihood of a GLM under latent exchangeability prior (LEAP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the latent exchangeability prior (LEAP).

The arguments related to MCMC sampling are utilized to draw samples from the LEAP. These samples are then used to compute the logarithm of the normalizing constant of the LEAP using historical data sets.

Usage

```
glm.logml.leap(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from glm.leap() giving posterior samples of a GLM under the latent exchangeability prior (LEAP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "glm_leap"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the latent exchangeability prior (LEAP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the LEAP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019   = actg019[1:100, ]
  actg036   = actg036[1:50, ]
  formula   = outcome ~ scale(age) + race + treatment + scale(cd4)
  family    = binomial('logit')
  data_list = list(currdata = actg019, histdata = actg036)
  d.leap    = glm.leap(
    formula = formula,
    family = family,
    data.list = data_list,
    K = 2,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  glm.logml.leap(
    post.samples = d.leap,
    bridge.args = list(silent = TRUE),
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

---

glm.logml.map *Log marginal likelihood of a GLM under meta-analytic predictive (MAP) prior*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the meta-analytic predictive (MAP) prior. The MAP prior is equivalent to the prior induced by the Bayesian hierarchical model (BHM).

The arguments related to MCMC sampling are utilized to draw samples from the MAP prior. These samples are then used to compute the logarithm of the normalizing constant of the BHM using only historical data sets.

Usage

```
glm.logml.map(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from glm.bhm() giving posterior samples of a GLM under the Bayesian hierarchical model (BHM), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "glm_bhm"

logml the estimated logarithm of the marginal likelihood of the meta-analytic predictive (MAP) prior

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the Bayesian hierarchical model (BHM) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the BHM using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  formula = outcome ~ scale(age) + race + treatment + scale(cd4)
  family = binomial('logit')
  data_list = list(currdata = actg019, histdata = actg036)
```

```
  d.bhm = glm.bhm(
    formula = formula,
    family = family,
    data.list = data_list,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  glm.logml.map(
    post.samples = d.bhm,
    bridge.args = list(silent = TRUE),
    chains = 1, iter_warmup = 1000, iter_sampling = 2000
  )
}
```

`glm.logml.napp` *Log marginal likelihood of a GLM under normalized asymptotic power prior (NAPP)*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the normalized asymptotic power prior (NAPP).

Usage

```
glm.logml.napp(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `glm.napp()` giving posterior samples of a GLM under the normalized asymptotic power prior (NAPP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "glm_napp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the normalized asymptotic power prior (NAPP)

References

Ibrahim, J. G., Chen, M., Gwon, Y., and Chen, F. (2015). The power prior: Theory and applications. Statistics in Medicine, 34(28), 3724–3749.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  data_list = list(currdata = actg019, histdata = actg036)
  formula = cd4 ~ treatment + age + race
  family = poisson('log')
  d.napp = glm.napp(
    formula = formula, family = family,
    data.list = data_list,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  glm.logml.napp(
    post.samples = d.napp,
    bridge.args = list(silent = TRUE)
  )
}
```

`glm.logml.npp` *Log marginal likelihood of a GLM under normalized power prior (NPP)*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the normalized power prior (NPP).

Usage

```
glm.logml.npp(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `glm.npp()` giving posterior samples of a GLM under the normalized power prior (NPP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than samples, log_posterior, data, lb, ub) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "glm_npp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the normalized power prior (NPP)

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if(requireNamespace("parallel")){
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]

  library(parallel)
  ncores    = 2
  data.list = list(data = actg019, histdata = actg036)
  formula   = cd4 ~ treatment + age + race
  family    = poisson()
  a0        = seq(0, 1, length.out = 11)
  if (instantiate::stan_cmdstan_exists()) {
    ## call created function
    ## wrapper to obtain log normalizing constant in parallel package
    logncfun = function(a0, ...){
      hdbayes::glm.npp.lognc(
        formula = formula, family = family, a0 = a0, histdata = data.list[[2]],
        ...
      )
    }

    cl = makeCluster(ncores)
    clusterSetRNGStream(cl, 123)
    clusterExport(cl, varlist = c('formula', 'family', 'data.list'))
    a0.lognc = parLapply(
      cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
      iter_sampling = 1000, chains = 1, refresh = 0
    )
    stopCluster(cl)
    a0.lognc = data.frame( do.call(rbind, a0.lognc) )
```

```
    ## sample from normalized power prior
    d.npp = glm.npp(
      formula = formula,
      family = family,
      data.list = data.list,
      a0.lognc = a0.lognc$a0,
      lognc = matrix(a0.lognc$lognc, ncol = 1),
      chains = 1, iter_warmup = 500, iter_sampling = 1000,
      refresh = 0
    )
    glm.logml.npp(
      post.samples = d.npp,
      bridge.args = list(silent = TRUE)
    )
  }
}
```

`glm.logml.post` *Log marginal likelihood of a GLM under a normal/half-normal prior*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the normal/half-normal prior.

Usage

```
glm.logml.post(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `glm.post()` giving posterior samples of a GLM under the normal/half-normal prior, with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "glm_post"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the normal/half-normal prior

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  actg019 = actg019[1:100, ]
  data.list = list(currdata = actg019)
  formula = cd4 ~ treatment + age + race
  family = poisson('log')
  d.post = glm.post(
    formula = formula, family = family,
    data.list = data.list,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  glm.logml.post(
    post.samples = d.post,
    bridge.args = list(silent = TRUE)
  )
}
```

`glm.logml.pp` *Log marginal likelihood of a GLM under power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a GLM under the power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the PP using only historical data sets.

Usage

```
glm.logml.pp(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from glm.pp() giving posterior samples of a GLM under the power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

If all of the power prior parameters (a_0's) are equal to zero, or if the posterior samples are obtained from using only one data set (the current data), then the function will return the same result as the output from glm.logml.post().

If at least one of the power prior parameters (a_0's) is non-zero, the function will return a list with the following objects

model "glm_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the power prior (PP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the PP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
```

```
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  data_list = list(currdata = actg019, histdata = actg036)
  formula = cd4 ~ treatment + age + race
  family = poisson('log')
  a0 = 0.5
  d.pp = glm.pp(
    formula = formula, family = family,
    data.list = data_list,
    a0.vals = a0,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  glm.logml.pp(
    post.samples = d.pp,
    bridge.args = list(silent = TRUE),
    chains = 1, iter_warmup = 1000, iter_sampling = 2000
  )
}
```

`glm.napp` *Posterior of normalized asymptotic power prior (NAPP)*

Description

Sample from the posterior distribution of a GLM using the normalized asymptotic power prior (NAPP) by Ibrahim et al. (2015) doi:10.1002/sim.6728.

Usage

```
glm.napp(
  formula,
  family,
  data.list,
  offset.list = NULL,
  a0.shape1 = 1,
  a0.shape2 = 1,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates.

family	an object of class family. See ?stats::family.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical datasets.
offset.list	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
a0.shape1	first shape parameter for the i.i.d. beta prior on a0 vector. When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
a0.shape2	second shape parameter for the i.i.d. beta prior on a0 vector. When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The normalized asymptotic power prior (NAPP) assumes that the regression coefficients and logarithm of the dispersion parameter are a multivariate normal distribution with mean equal to the maximum likelihood estimate of the historical data and covariance matrix equal to a_0^{-1} multiplied by the inverse Fisher information matrix of the historical data, where a_0 is the power prior parameter (treated as random).

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Ibrahim, J. G., Chen, M., Gwon, Y., and Chen, F. (2015). The power prior: Theory and applications. Statistics in Medicine, 34(28), 3724–3749.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  data_list = list(currdata = actg019, histdata = actg036)
  glm.napp(
    formula = cd4 ~ treatment + age + race,
    family = poisson('log'),
    data.list = data_list,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

`glm.npp` *Posterior of normalized power prior (NPP)*

Description

Sample from the posterior distribution of a GLM using the normalized power prior (NPP) by Duan et al. (2006) doi:10.1002/env.752.

Usage

```
glm.npp(
  formula,
  family,
  data.list,
  a0.lognc,
  lognc,
  offset.list = NULL,
  beta.mean = NULL,
  beta.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  a0.shape1 = 1,
  a0.shape2 = 1,
  a0.lower = NULL,
  a0.upper = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets.
a0.lognc	a vector giving values of the power prior parameter for which the logarithm of the normalizing constant has been evaluated.
lognc	an S by T matrix where S is the length of a0.lognc, T is the number of historical data sets, and the j-th column, j = 1, ..., T, is a vector giving the logarithm of the normalizing constant (as estimated by glm.npp.lognc() for a0.lognc using the j-th historical data set.
offset.list	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
disp.mean	location parameter for the half-normal prior on dispersion parameter. Defaults to 0.
disp.sd	scale parameter for the half-normal prior on dispersion parameter. Defaults to 10.
a0.shape1	first shape parameter for the i.i.d. beta prior on a_0 vector. When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
a0.shape2	second shape parameter for the i.i.d. beta prior on a_0 vector. When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
a0.lower	a scalar or a vector whose dimension is equal to the number of historical data sets giving the lower bounds for each element of the a_0 vector. If a scalar is provided, a0.lower will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
a0.upper	a scalar or a vector whose dimension is equal to the number of historical data sets giving the upper bounds for each element of the a_0 vector. If a scalar is provided, same as for a0.lower. Defaults to a vector of 1s.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.

chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

Before using this function, users must estimate the logarithm of the normalizing constant across a range of different values for the power prior parameter (a_0), possibly smoothing techniques over a find grid. The power prior parameters (a_0's) are treated as random with independent beta priors. The initial priors on the regression coefficients are independent normal priors. The current and historical data sets are assumed to have a common dispersion parameter with a half-normal prior (if applicable). For normal linear models, the exact normalizing constants for NPP can be computed. See the implementation in lm.npp().

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

See Also

glm.npp.lognc()

Examples

```
if(requireNamespace("parallel")){
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]

  library(parallel)
  ncores    = 2
  data.list = list(data = actg019, histdata = actg036)
  formula   = cd4 ~ treatment + age + race
  family    = poisson()
  a0        = seq(0, 1, length.out = 11)
  if (instantiate::stan_cmdstan_exists()) {
    ## call created function
    ## wrapper to obtain log normalizing constant in parallel package
    logncfun = function(a0, ...){
```

```
      hdbayes::glm.npp.lognc(
        formula = formula, family = family, a0 = a0, histdata = data.list[[2]],
        ...
      )
    }

    cl = makeCluster(ncores)
    clusterSetRNGStream(cl, 123)
    clusterExport(cl, varlist = c('formula', 'family', 'data.list'))
    a0.lognc = parLapply(
      cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
      iter_sampling = 1000, chains = 1, refresh = 0
    )
    stopCluster(cl)
    a0.lognc = data.frame( do.call(rbind, a0.lognc) )

    ## sample from normalized power prior
    glm.npp(
      formula = formula,
      family = family,
      data.list = data.list,
      a0.lognc = a0.lognc$a0,
      lognc = matrix(a0.lognc$lognc, ncol = 1),
      chains = 1, iter_warmup = 500, iter_sampling = 1000,
      refresh = 0
    )
  }
}
```

`glm.npp.lognc` *Estimate the logarithm of the normalizing constant for normalized power prior (NPP) for one data set*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the normalizing constant for the NPP for a fixed value of the power prior parameter $a_0 \in (0, 1)$ for one data set. The initial priors are independent normal priors on the regression coefficients and a half-normal prior on the dispersion parameter (if applicable).

Usage

```
glm.npp.lognc(
  formula,
  family,
  histdata,
  a0,
  offset0 = NULL,
```

```
  beta.mean = NULL,
  beta.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.
histdata	a data.frame giving the historical data.
a0	the power prior parameter (a scalar between 0 and 1).
offset0	vector whose dimension is equal to the rows of the historical data set giving an offset for the historical data. Defaults to a vector of 0s.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the normal initial prior on regression coefficients given the dispersion parameter. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. The sd used is sqrt(dispersion) * beta.sd. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
disp.mean	location parameter for the half-normal prior on dispersion parameter. Defaults to 0.
disp.sd	scale parameter for the half-normal prior on dispersion parameter. Defaults to 10.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g. seed, refresh, init).

Value

The function returns a vector giving the value of a0, the estimated logarithm of the normalizing constant, the minimum estimated bulk effective sample size of the MCMC sampling, and the maximum Rhat.

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg036)
  ## take subset for speed purposes
  actg036 = actg036[1:50, ]
  glm.npp.lognc(
    cd4 ~ treatment + age + race,
    family = poisson(), histdata = actg036, a0 = 0.5,
    chains = 1, iter_warmup = 500, iter_sampling = 5000
  )
}
```

---

glm.post *Posterior of a normal/half-normal prior*

Description

Sample from the posterior distribution of a GLM using a normal/half-normal prior.

Usage

```
glm.post(
  formula,
  family,
  data.list,
  offset.list = NULL,
  beta.mean = NULL,
  beta.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.
data.list	a list consisting of one data.frame giving the current data. If data.list has more than one data.frame, only the first element will be used as the current data.
offset.list	a list consisting of one vector giving the offset for the current data. The length of the vector is equal to the number of rows in the current data. The vector has all values set to 0 by default. If offset.list has more than one vector, same as for data.list.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the normal prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the normal prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
disp.mean	location parameter for the half-normal prior on dispersion parameter. Defaults to 0. If disp.mean is a vector with length > 1, only the first element will be used as disp.mean.
disp.sd	scale parameter for the half-normal prior on dispersion parameter. Defaults to 10. If disp.sd is a vector with length > 1, same as for disp.mean.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The priors on the regression coefficients are independent normal distributions. When the normal priors are elicited with large variances, the prior is also referred to as the reference or vague prior. The dispersion parameter is assumed to be independent of the regression coefficients with a half-normal prior (if applicable).

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  data.list = list(currdata = actg019)
  glm.post(
    formula = cd4 ~ treatment + age + race,
    family = poisson('log'),
    data.list = data.list,
    beta.sd   = 10,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

---

glm.pp *Posterior of power prior (PP) with fixed a_0*

Description

Sample from the posterior distribution of a GLM using the power prior (PP) by Ibrahim and Chen (2000) doi:10.1214/ss/1009212673.

Usage

```
glm.pp(
  formula,
  family,
  data.list,
  a0.vals,
  offset.list = NULL,
  beta.mean = NULL,
  beta.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.

data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets.
a0.vals	a scalar between 0 and 1 or a vector whose dimension is equal to the number of historical data sets giving the (fixed) power prior parameter for each historical data set. Each element of vector should be between 0 and 1. If a scalar is provided, same as for beta.mean.
offset.list	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
disp.mean	location parameter for the half-normal prior on dispersion parameter. Defaults to 0.
disp.sd	scale parameter for the half-normal prior on dispersion parameter. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The power prior parameters (a_0's) are treated as fixed. The initial priors on the regression coefficients are independent normal priors. The current and historical data sets are assumed to have a common dispersion parameter with a half-normal prior (if applicable).

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  ## take subset for speed purposes
  actg019 = actg019[1:100, ]
  actg036 = actg036[1:50, ]
  data_list = list(currdata = actg019, histdata = actg036)
  glm.pp(
    formula = cd4 ~ treatment + age + race,
    family = poisson('log'),
    data.list = data_list,
    a0.vals = 0.5,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

`glm.rmap` *Posterior of robust meta-analytic predictive prior (RMAP)*

Description

Sample from the posterior distribution of a GLM using the robust meta-analytic predictive prior (RMAP) by Schmidli et al. (2014) doi:10.1111/biom.12242.

Usage

```
glm.rmap(
  formula,
  family,
  data.list,
  offset.list = NULL,
  w = 0.1,
  meta.mean.mean = NULL,
  meta.mean.sd = NULL,
  meta.sd.mean = NULL,
  meta.sd.sd = NULL,
  disp.mean = NULL,
  disp.sd = NULL,
  norm.vague.mean = NULL,
  norm.vague.sd = NULL,
  bridge.args = NULL,
  iter_warmup = 1000,
```

```
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates.
family	an object of class family. See ?stats::family.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets.
offset.list	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
w	a scalar between 0 and 1 giving how much weight to put on the historical data. Defaults to 0.1.
meta.mean.mean	same as meta.mean.mean in glm.bhm(). It is a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the normal hyperpriors on the mean hyperparameters of regression coefficients in Bayesian hierarchical model (BHM). If a scalar is provided, meta.mean.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
meta.mean.sd	same as meta.mean.sd in glm.bhm(). It is a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the normal hyperpriors on the mean hyperparameters of regression coefficients in BHM. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 10s.
meta.sd.mean	same as meta.sd.mean in glm.bhm(). It is a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the half-normal hyperpriors on the sd hyperparameters of regression coefficients in BHM. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 0s.
meta.sd.sd	same as meta.sd.sd in glm.bhm(). It is a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the half-normal hyperpriors on the sd hyperparameters of regression coefficients in BHM. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 1s.
disp.mean	a scalar or a vector whose dimension is equal to the number of data sets (including the current data) giving the location parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 0s.
disp.sd	a scalar or a vector whose dimension is equal to the number of data sets (including the current data) giving the scale parameters for the half-normal priors on the dispersion parameters. If a scalar is provided, same as for meta.mean.mean. Defaults to a vector of 10s.

`norm.vague.mean`
: same as `beta.mean` in `glm.post()`. It is a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the vague normal prior on regression coefficients. If a scalar is provided, `norm.vague.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

`norm.vague.sd`
: same as `beta.sd` in `glm.post()`. It is a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the vague normal prior on regression coefficients. If a scalar is provided, same as for `norm.vague.mean`. Defaults to a vector of 10s.

`bridge.args`
: a `list` giving arguments (other than samples, log_posterior, data, lb, ub) to pass onto `bridgesampling::bridge_sampler()`.

`iter_warmup`
: number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling`
: number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains`
: number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...`
: arguments passed to `sample()` method in cmdstanr package (e.g. seed, refresh, init).

Details

The robust meta-analytic predictive prior (RMAP) is a two-part mixture prior consisting of a meta-analytic predictive (MAP) prior (the prior induced by Bayesian hierarchical model (BHM)) and a vague (i.e., non-informative) prior (specifically, the normal/half-normal prior with large variances). Although Schmidli et al. (2014) recommends to use a finite mixture of conjugate priors to approximate the BHM, it can be difficult and time-consuming to come up with an appropriate approximation.

Instead, the approach taken by hdbayes is to use the marginal likelihood of the MAP and vague priors. Specifically, note that the posterior distribution of a GLM under RMAP is also a two-part mixture distribution. The updated mixture weight for posterior density under the MAP prior is

$$\widetilde{w} = \frac{wZ_I(D, D_0)}{wZ_I(D, D_0) + (1-w)Z_V(D)},$$

where w is the prior mixture weight for the MAP prior in RMAP, $Z_I(D, D_0)$ is the marginal likelihood of the MAP prior, and $Z_V(D)$ is the marginal likelihood of the vague prior.

Value

The function returns a `list` with the following objects

post.samples an object of class `draws_df` giving posterior samples under the robust meta-analytic predictive prior (RMAP)

post.weight.bhm a scalar between 0 and 1 giving the updated mixture weight for posterior density under the MAP prior

post.samples.bhm an object of class `draws_df` giving posterior samples under the Bayesian hierarchical model (BHM) (equivalently, the MAP prior), obtained from using `glm.bhm()`

post.samples.vague an object of class `draws_df` giving posterior samples under the vague/non-informative prior, obtained from using `glm.post()`

bs.map output from computing log marginal likelihood of the prior induced by the BHM (referred to as the meta-analytic predictive (MAP) prior) via `glm.logml.map()` function

bs.vague output from computing log marginal likelihood of the vague prior via `glm.logml.post()` function

References

Schmidli, H., Gsteiger, S., Roychoudhury, S., O'Hagan, A., Spiegelhalter, D., and Neuenschwander, B. (2014). Robust meta-analytic-predictive priors in clinical trials with historical control information. Biometrics, 70(4), 1023–1032.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An R package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019) ## current data
  data(actg036) ## historical data
  ## take subset for speed purposes
  actg019 = actg019[1:150, ]
  actg036 = actg036[1:100, ]
  data.list = list(actg019, actg036)
  glm.rmap(
    formula = outcome ~ scale(age) + race + treatment + scale(cd4),
    family = binomial('logit'),
    data.list = data.list,
    w = 0.1,
    chains = 1, iter_warmup = 1000, iter_sampling = 2000
  )
}
```

IBCSG_curr *International Breast Cancer Study Group (IBCSG) Trial VI Data*

Description

A data set from the IBCSG Trial VI investigating both the duration of adjuvant chemotherapy (3 versus 6 initial cycles of oral cyclophosphamide, methotrexate, and fluorouracil (CMF)) and the reintroduction of single courses of delayed chemotherapy in node-positive premenopausal breast cancer patients. The study results were described by IBCSG (1996) doi:10.1200/JCO.1996.14.6.1885 and Hürny et al. (1992) doi:10.1016/0959-8049(92)90399-m. This data set only includes patients above the age of 40 (i.e., age $\geq$ 40) and treats the measurements of patients'

physical well-being on month 18 as the outcome. The IBCSG_hist data set includes patients from the same study but with age < 40. We can use the IBCSG_hist data as the historical data and the IBCSG_curr data as the current data.

Usage

```
IBCSG_curr
```

Format

A data frame with 488 rows and 8 variables:

phys18 outcome variable, integer scores between 0 and 100 measuring the patients' physical well-being on month 18, with a higher score indicating a better physical well-being

phys1 physical well-being scores assessed at the start of the study

n_init_cycles number of initial cycles of CMF, equal to 3 or 6

reintroduction indicator of reintroduction of chemotherapy, 0 = no reintroduction, 1 = having reintroduction

age patient age in years

country country, ANZ = New Zealand/Australia, CH = Switzerland, SWED = Sweden

nodegp indicator of number of positive nodes being greater than or equal to 4, 0 = less than 4, 1 = 4+

ER estrogen receptor (ER) status indicator, 0 = negative, 1 = positive

References

International Breast Cancer Study Group. (1996). Duration and reintroduction of adjuvant chemotherapy for node-positive premenopausal breast cancer patients. Journal of Clinical Oncology, 14(6), 1885–1894.

Hürny, C., Bernhard, J., Gelber, R. D., Coates, A., Castiglione, M., Isley, M., Dreher, D., Peterson, H., Goldhirsch, A., and Senn, H.-J. (1992). Quality of life measures for patients receiving adjuvant therapy for breast cancer: An international trial. European Journal of Cancer, 28(1), 118–124.

Chi, Y.-Y. and Ibrahim, J. G. (2005). Joint models for multivariate longitudinal and Multivariate Survival Data. Biometrics, 62(2), 432–445.

IBCSG_hist *International Breast Cancer Study Group (IBCSG) Trial VI Data*

Description

A data set from the IBCSG Trial VI investigating both the duration of adjuvant chemotherapy (3 versus 6 initial cycles of oral cyclophosphamide, methotrexate, and fluorouracil (CMF)) and the reintroduction of single courses of delayed chemotherapy in node-positive premenopausal breast cancer patients. The study results were described by IBCSG (1996) doi:10.1200/JCO.1996.14.6.1885 and Hürny et al. (1992) doi:10.1016/0959-8049(92)90399-m. This data set only includes patients under the age of 40 (i.e., age < 40) and treats the measurements of patients' physical well-being on month 18 as the outcome. The IBCSG_curr data set includes patients from the same study but with age $\geq$ 40. We can use the IBCSG_hist data as the historical data and the IBCSG_curr data as the current data.

Usage

```
IBCSG_hist
```

Format

A data frame with 103 rows and 8 variables:

phys18 outcome variable, integer scores between 0 and 100 measuring the patients' physical well-being on month 18, with a higher score indicating a better physical well-being

phys1 physical well-being scores assessed at the start of the study

n_init_cycles number of initial cycles of CMF, equal to 3 or 6

reintroduction indicator of reintroduction of chemotherapy, 0 = no reintroduction, 1 = having reintroduction

age patient age in years

country country, ANZ = New Zealand/Australia, CH = Switzerland, SWED = Sweden

nodegp indicator of number of positive nodes being greater than or equal to 4, 0 = less than 4, 1 = 4+

ER estrogen receptor (ER) status indicator, 0 = negative, 1 = positive

References

International Breast Cancer Study Group. (1996). Duration and reintroduction of adjuvant chemotherapy for node-positive premenopausal breast cancer patients. Journal of Clinical Oncology, 14(6), 1885–1894.

Hürny, C., Bernhard, J., Gelber, R. D., Coates, A., Castiglione, M., Isley, M., Dreher, D., Peterson, H., Goldhirsch, A., and Senn, H.-J. (1992). Quality of life measures for patients receiving adjuvant therapy for breast cancer: An international trial. European Journal of Cancer, 28(1), 118–124.

Chi, Y.-Y. and Ibrahim, J. G. (2005). Joint models for multivariate longitudinal and Multivariate Survival Data. Biometrics, 62(2), 432–445.

`lm.npp` *Posterior of normalized power prior (NPP) for normal linear models*

Description

Sample from the posterior distribution of a normal linear model using the NPP by Duan et al. (2006) doi:10.1002/env.752. The power prior parameters (a_0's) are treated as random with independent beta priors. The current and historical data sets are assumed to have a common dispersion parameter (σ^2) with an inverse-gamma prior. Conditional on σ^2, the initial priors on the regression coefficients are independent normal distributions with variance $\propto (\sigma^2)^{-1}$. In this case, the normalizing constant for the NPP has a closed form.

Usage

```
lm.npp(
  formula,
  data.list,
  offset.list = NULL,
  beta.mean = NULL,
  beta.sd = NULL,
  sigmasq.shape = 2.1,
  sigmasq.scale = 1.1,
  a0.shape1 = 1,
  a0.shape2 = 1,
  a0.lower = NULL,
  a0.upper = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`formula`	a two-sided formula giving the relationship between the response variable and covariates.
`data.list`	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets.
`offset.list`	a list of vectors giving the offsets for each data. The length of offset.list is equal to the length of data.list. The length of each element of offset.list is equal to the number of rows in the corresponding element of data.list. Defaults to a list of vectors of 0s.
`beta.mean`	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients. Conditional on the variance parameter sigmasq for the outcome, beta.sd * sqrt(sigmasq) gives the sd for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
sigmasq.shape	shape parameter for inverse-gamma prior on variance parameter. Defaults to 2.1.
sigmasq.scale	scale parameter for inverse-gamma prior on variance parameter. Defaults to 1.1.
a0.shape1	first shape parameter for the i.i.d. beta prior on a0 vector. When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
a0.shape2	second shape parameter for the i.i.d. beta prior on a0 vector. When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.
a0.lower	a scalar or a vector whose dimension is equal to the number of historical data sets giving the lower bounds for each element of the a0 vector. If a scalar is provided, a0.lower will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
a0.upper	a scalar or a vector whose dimension is equal to the number of historical data sets giving the upper bounds for each element of the a0 vector. If a scalar is provided, same as for a0.lower. Defaults to a vector of 1s.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g. seed, refresh, init).

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  data(actg019)
  data(actg036)
  data_list = list(currdata = actg019, histdata = actg036)
```

```
  lm.npp(
    formula = cd4 ~ treatment + age + race,
    data.list = data_list,
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
}
```

---

pwe.bhm *Posterior of Bayesian hierarchical model (BHM)*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard) using the Bayesian hierarchical model (BHM).

Usage

```
pwe.bhm(
  formula,
  data.list,
  breaks,
  meta.mean.mean = NULL,
  meta.mean.sd = NULL,
  meta.sd.mean = NULL,
  meta.sd.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting piecewise exponential (PWE) models, all historical data sets will be stacked into one historical data set.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

`meta.mean.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, `meta.mean.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

`meta.mean.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the normal hyperpriors on the mean hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 10s.

`meta.sd.mean` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the means for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 0s.

`meta.sd.sd` a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sds for the half-normal hyperpriors on the sd hyperparameters of regression coefficients. If a scalar is provided, same as for `meta.mean.mean`. Defaults to a vector of 1s.

`base.hazard.mean` a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `meta.mean.mean`. Defaults to 0.

`base.hazard.sd` a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `meta.mean.mean`. Defaults to 10.

`get.loglik` whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup` number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling` number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains` number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...` arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The Bayesian hierarchical model (BHM) assumes that the regression coefficients for the historical and current data are different, but are correlated through a common distribution, whose hyperparameters (i.e., mean and standard deviation (sd) (the covariance matrix is assumed to have a diagonal structure)) are treated as random. The number of regression coefficients for the current data is assumed to be the same as that for the historical data.

The hyperpriors on the mean and the sd hyperparameters are independent normal and independent half-normal distributions, respectively. The baseline hazard parameters for both current and historical data models are assumed to be independent and identically distributed, each assigned a half-normal prior.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.bhm(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`pwe.commensurate` *Posterior of commensurate prior (CP)*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard) using the commensurate prior (CP) by Hobbs et al. (2011) doi:10.1111/j.1541-0420.2011.01564.x.

Usage

```
pwe.commensurate(
  formula,
  data.list,
  breaks,
  beta0.mean = NULL,
  beta0.sd = NULL,
  p.spike = 0.1,
  spike.mean = 200,
  spike.sd = 0.1,
  slab.mean = 0,
  slab.sd = 5,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting piecewise exponential (PWE) models, all historical data sets will be stacked into one historical data set.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
beta0.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the prior on the historical data regression coefficients. If a scalar is provided, `beta0.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta0.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the prior on the historical data regression coefficients. If a scalar is provided, same as for `beta0.mean`. Defaults to a vector of 10s.
p.spike	a scalar between 0 and 1 giving the probability of the spike component in spike-and-slab prior on commensurability parameter τ. Defaults to 0.1.
spike.mean	a scalar giving the location parameter for the half-normal prior (spike component) on τ. Defaults to 200.
spike.sd	a scalar giving the scale parameter for the half-normal prior (spike component) on τ. Defaults to 0.1.

`slab.mean` a scalar giving the location parameter for the half-normal prior (slab component) on τ. Defaults to 0.

`slab.sd` a scalar giving the scale parameter for the half-normal prior (slab component) on τ. Defaults to 5.

`base.hazard.mean` a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta0.mean`. Defaults to 0.

`base.hazard.sd` a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta0.mean`. Defaults to 10.

`get.loglik` whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup` number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling` number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains` number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...` arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The commensurate prior (CP) assumes that the regression coefficients for the current data model conditional on those for the historical data model are independent normal distributions with mean equal to the corresponding regression coefficients for the historical data and variance equal to the inverse of the corresponding elements of a vector of precision parameters (referred to as the commensurability parameter τ). We regard τ as random and elicit a spike-and-slab prior, which is specified as a mixture of two half-normal priors, on τ.

The number of current data regression coefficients is assumed to be the same as that of historical data regression coefficients. The baseline hazard parameters for both current and historical data models are assumed to be independent and identically distributed, each assigned a half-normal prior.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      p.spike = 0.1,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`pwe.leap` *Posterior of latent exchangeability prior (LEAP)*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard) using the latent exchangeability prior (LEAP) by Alt et al. (2024) doi:10.1093/biomtc/ujae083.

Usage

```
pwe.leap(
  formula,
  data.list,
  breaks,
  K = 2,
  prob.conc = NULL,
```

```
  gamma.lower = 0,
  gamma.upper = 1,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting piecewise exponential (PWE) models, all historical data sets will be stacked into one historical data set.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
K	the desired number of classes to identify. Defaults to 2.
prob.conc	a scalar or a vector of length `K` giving the concentration parameters for Dirichlet prior. If length == 2, a `Beta(prob.conc[1], prob.conc[2])` prior is used. If a scalar is provided, `prob.conc` will be a vector of repeated elements of the given scalar. Defaults to a vector of 1s.
gamma.lower	a scalar giving the lower bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 0.
gamma.upper	a scalar giving the upper bound for probability of subjects in historical data being exchangeable with subjects in current data. Defaults to 1.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
base.hazard.mean	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.

base.hazard.sd	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The latent exchangeability prior (LEAP) discounts the historical data by identifying the most relevant individuals from the historical data. It is equivalent to a prior induced by the posterior of a finite mixture model for the historical data set.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
```

```
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.leap(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      K = 2,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`pwe.logml.commensurate` *Log marginal likelihood of a piecewise exponential (PWE) model under the commensurate prior (CP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the commensurate prior (CP).

The arguments related to MCMC sampling are utilized to draw samples from the commensurate prior. These samples are then used to compute the logarithm of the normalizing constant of the commensurate prior using historical data sets.

Usage

```
pwe.logml.commensurate(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `pwe.commensurate()` giving posterior samples of a PWE model under the commensurate prior (CP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "pwe_commensurate"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the commensurate prior (CP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the CP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Hobbs, B. P., Carlin, B. P., Mandrekar, S. J., and Sargent, D. J. (2011). Hierarchical commensurate and power prior models for adaptive incorporation of historical information in clinical trials. Biometrics, 67(3), 1047–1056.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
```

```
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.cp = pwe.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      p.spike = 0.1,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    pwe.logml.commensurate(
      post.samples = d.cp,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`pwe.logml.leap` *Log marginal likelihood of a piecewise exponential (PWE) model under latent exchangeability prior (LEAP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the latent exchangeability prior (LEAP).

The arguments related to MCMC sampling are utilized to draw samples from the LEAP. These samples are then used to compute the logarithm of the normalizing constant of the LEAP using historical data sets.

Usage

```
pwe.logml.leap(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `pwe.leap()` giving posterior samples of a PWE model under the latent exchangeability prior (LEAP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.

bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "pwe_leap"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the latent exchangeability prior (LEAP) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the LEAP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Alt, E. M., Chang, X., Jiang, X., Liu, Q., Mo, M., Xia, H. M., and Ibrahim, J. G. (2024). LEAP: The latent exchangeability prior for borrowing information from historical data. Biometrics, 80(3).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
```

```
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.leap = pwe.leap(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      K = 2,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    pwe.logml.leap(
      post.samples = d.leap,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`pwe.logml.map` *Log marginal likelihood of a piecewise exponential (PWE) model under the meta-analytic predictive (MAP) prior*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the meta-analytic predictive (MAP) prior. The MAP prior is equivalent to the prior induced by the Bayesian hierarchical model (BHM).

The arguments related to MCMC sampling are utilized to draw samples from the MAP prior. These samples are then used to compute the logarithm of the normalizing constant of the MAP prior using only historical data set.

Usage

```
pwe.logml.map(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

post.samples	output from pwe.bhm() giving posterior samples of a PWE model under the Bayesian hierarchical model (BHM), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
bridge.args	a list giving arguments (other than samples, log_posterior, data, lb, and ub) to pass onto bridgesampling::bridge_sampler().
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a list with the following objects

model "pwe_bhm"

logml the estimated logarithm of the marginal likelihood

bs an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the Bayesian hierarchical model (BHM) using all data sets

bs.hist an object of class bridge or bridge_list containing the output from using bridgesampling::bridge_sampler() to compute the logarithm of the normalizing constant of the BHM using historical data set

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
```

```
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.bhm = pwe.bhm(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    pwe.logml.map(
      post.samples = d.bhm,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`pwe.logml.npp` *Log marginal likelihood of a piecewise exponential (PWE) model under normalized power prior (NPP)*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the normalized power prior (NPP).

Usage

```
pwe.logml.npp(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `pwe.npp()` giving posterior samples of a PWE model under the normalized power prior (NPP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "pwe_npp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the normalized power prior (NPP)

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if(requireNamespace("parallel")){
  library(parallel)
  ncores    = 2

  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin
  }

  a0 = seq(0, 1, length.out = 11)
  if (instantiate::stan_cmdstan_exists()) {
    ## call created function
    ## wrapper to obtain log normalizing constant in parallel package
    logncfun = function(a0, ...){
      hdbayes::pwe.npp.lognc(
```

```
      formula = formula, histdata = data_list[[2]], breaks = breaks, a0 = a0,
      ...
    )
  }

  cl = makeCluster(ncores)
  clusterSetRNGStream(cl, 123)
  clusterExport(cl, varlist = c('formula', 'data_list', 'breaks'))
  a0.lognc = parLapply(
    cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
    iter_sampling = 1000, chains = 1, refresh = 0
  )
  stopCluster(cl)
  a0.lognc = data.frame( do.call(rbind, a0.lognc) )

  ## sample from normalized power prior
  d.npp = pwe.npp(
    formula = formula,
    data.list = data_list,
    a0.lognc = a0.lognc$a0,
    lognc = a0.lognc$lognc,
    breaks = breaks,
    chains = 1, iter_warmup = 500, iter_sampling = 1000,
    refresh = 0
  )
  pwe.logml.npp(
    post.samples = d.npp,
    bridge.args = list(silent = TRUE)
  )
 }
}
```

`pwe.logml.post` *Log marginal likelihood of a piecewise exponential (PWE) model under a normal/half-normal prior*

Description

Uses bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the normal/half-normal prior.

Usage

```
pwe.logml.post(post.samples, bridge.args = NULL)
```

Arguments

`post.samples`	output from `pwe.post()` giving posterior samples of a PWE model under the normal/half-normal prior, with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.

`bridge.args` a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

Value

The function returns a `list` with the following objects

model "pwe_post"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the marginal likelihood of the PWE model under the normal/half-normal prior

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1690)
    ## take subset for speed purposes
    E1690 = E1690[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.post = pwe.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    pwe.logml.post(
      post.samples = d.post,
      bridge.args = list(silent = TRUE)
    )
  }
}
```

`pwe.logml.pp` *Log marginal likelihood of a piecewise exponential (PWE) model under the power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the PP using only historical data set.

Usage

```
pwe.logml.pp(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `pwe.pp()` giving posterior samples of a PWE model under the power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.
`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Value

If the power prior parameter (a_0) is equal to zero, then the function will return the same result as the output from `pwe.logml.post()`.

If the power prior parameters (a_0) is non-zero, the function will return a `list` with the following objects

model "pwe_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the power prior (PP) using all data sets

bs.hist an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the PP using historical data set

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.pp = pwe.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      a0 = 0.5,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
    pwe.logml.pp(
      post.samples = d.pp,
      bridge.args = list(silent = TRUE),
```

```
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
  }
}
```

`pwe.logml.stratified.pp` *Log marginal likelihood of a piecewise exponential (PWE) model under the stratified power prior (PP)*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the marginal likelihood of a PWE model under the stratified power prior (PP).

The arguments related to MCMC sampling are utilized to draw samples from the stratified power prior (PP). These samples are then used to compute the logarithm of the normalizing constant of the stratified PP using only historical data sets.

Usage

```
pwe.logml.stratified.pp(
  post.samples,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

`post.samples`	output from `pwe.stratified.pp()` giving posterior samples of a PWE model under the stratified power prior (PP), with an attribute called 'data' which includes the list of variables specified in the data block of the Stan program.
`bridge.args`	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.
`iter_warmup`	number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.
`iter_sampling`	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.
`chains`	number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.
`...`	arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Value

The function returns a `list` with the following objects

model "pwe_stratified_pp"

logml the estimated logarithm of the marginal likelihood

bs an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the stratified power prior (PP) using all data sets

bs.hist an object of class `bridge` or `bridge_list` containing the output from using `bridgesampling::bridge_sampler()` to compute the logarithm of the normalizing constant of the stratified PP using historical data sets

min_ess_bulk the minimum estimated bulk effective sample size of the MCMC sampling

max_Rhat the maximum Rhat

References

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y., & Yue, L. Q. (2019). Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. Journal of Biopharmaceutical Statistics, 29(5), 731–748.

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    data_list = list(currdata = E1690, histdata = E1684)
    strata_list = list(rep(1:2, each = 25), rep(1:2, each = 50))
    # Alternatively, we can determine the strata based on propensity scores
    # using the psrwe package, which is available on GitHub
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    d.stratified.pp = pwe.stratified.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment,
      data.list = data_list,
```

```
    strata.list = strata_list,
    breaks = breaks,
    a0.strata = c(0.3, 0.5),
    chains = 1, iter_warmup = 500, iter_sampling = 1000
  )
  pwe.logml.stratified.pp(
    post.samples = d.stratified.pp,
    bridge.args = list(silent = TRUE),
    chains = 1, iter_warmup = 1000, iter_sampling = 2000
  )
 }
}
```

---

pwe.npp *Posterior of normalized power prior (NPP)*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model using the normalized power prior (NPP) by Duan et al. (2006) doi:10.1002/env.752.

Usage

```
pwe.npp(
  formula,
  data.list,
  a0.lognc,
  lognc,
  breaks,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  a0.shape1 = 1,
  a0.shape2 = 1,
  a0.lower = 0,
  a0.upper = 1,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the survival::Surv(time,

event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.

data.list — a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For fitting piecewise exponential (PWE) models, all historical data sets will be stacked into one historical data set.

a0.lognc — a vector giving values of the power prior parameter for which the logarithm of the normalizing constant has been evaluated.

lognc — a vector giving the logarithm of the normalizing constant (as estimated by pwe.npp.lognc() for each value of a0.lognc using the historical data set.

breaks — a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.

beta.mean — a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.

beta.sd — a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.

base.hazard.mean — a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 0.

base.hazard.sd — a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 10.

a0.shape1 — first shape parameter for the beta prior on the power prior parameter (a_0). When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.

a0.shape2 — second shape parameter for the beta prior on the power prior parameter (a_0). When a0.shape1 == 1 and a0.shape2 == 1, a uniform prior is used.

a0.lower — a scalar giving the lower bound for a_0. Defaults to 0.

a0.upper — a scalar giving the upper bound for a_0. Defaults to 1.

get.loglik — whether to generate log-likelihood matrix. Defaults to FALSE.

iter_warmup — number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.

iter_sampling — number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.

chains — number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.

... — arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

Before using this function, users must estimate the logarithm of the normalizing constant across a range of different values for the power prior parameter (a_0), possibly smoothing techniques over a find grid. The power prior parameters (a_0's) are treated as random with independent beta priors. The initial priors on the regression coefficients are independent normal priors. The current and historical data models are assumed to share the baseline hazard parameters with half-normal priors.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Duan, Y., Ye, K., and Smith, E. P. (2005). Evaluating water quality using power priors to incorporate historical information. Environmetrics, 17(1), 95–106.

See Also

`pwe.npp.lognc()`

Examples

```
if(requireNamespace("parallel")){
  library(parallel)
  ncores    = 2

  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin
```

```
  }

  a0 = seq(0, 1, length.out = 11)
  if (instantiate::stan_cmdstan_exists()) {
    ## call created function
    ## wrapper to obtain log normalizing constant in parallel package
    logncfun = function(a0, ...){
      hdbayes::pwe.npp.lognc(
        formula = formula, histdata = data_list[[2]], breaks = breaks, a0 = a0,
        ...
      )
    }

    cl = makeCluster(ncores)
    clusterSetRNGStream(cl, 123)
    clusterExport(cl, varlist = c('formula', 'data_list', 'breaks'))
    a0.lognc = parLapply(
      cl = cl, X = a0, fun = logncfun, iter_warmup = 500,
      iter_sampling = 1000, chains = 1, refresh = 0
    )
    stopCluster(cl)
    a0.lognc = data.frame( do.call(rbind, a0.lognc) )

    ## sample from normalized power prior
    pwe.npp(
      formula = formula,
      data.list = data_list,
      a0.lognc = a0.lognc$a0,
      lognc = a0.lognc$lognc,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000,
      refresh = 0
    )
  }
}
```

`pwe.npp.lognc` *Estimate the logarithm of the normalizing constant for normalized power prior (NPP) for one data set*

Description

Uses Markov chain Monte Carlo (MCMC) and bridge sampling to estimate the logarithm of the normalizing constant of a piecewise exponential (PWE) model under the NPP for a fixed value of the power prior parameter $a_0 \in (0, 1)$ for one data set. The initial priors are independent normal priors on the regression coefficients and half-normal priors on the baseline hazard parameters.

Usage

```
pwe.npp.lognc(
  formula,
  histdata,
  breaks,
  a0,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  bridge.args = NULL,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
histdata	a `data.frame` giving the historical data.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
a0	a scalar between 0 and 1 giving the (fixed) power prior parameter for the historical data.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
base.hazard.mean	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.
base.hazard.sd	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.
bridge.args	a `list` giving arguments (other than `samples`, `log_posterior`, `data`, `lb`, and `ub`) to pass onto `bridgesampling::bridge_sampler()`.

iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Value

The function returns a vector giving the value of a0, the estimated logarithm of the normalizing constant, the minimum estimated bulk effective sample size of the MCMC sampling, and the maximum Rhat.

References

Gronau, Q. F., Singmann, H., and Wagenmakers, E.-J. (2020). bridgesampling: An r package for estimating normalizing constants. Journal of Statistical Software, 92(10).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1684[E1684$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.npp.lognc(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      histdata = E1684,
      breaks = breaks,
      a0 = 0.5,
      bridge.args = list(silent = TRUE),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

pwe.post *Posterior of a normal/half-normal prior*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard) using a normal/half-normal prior.

Usage

```
pwe.post(
  formula,
  data.list,
  breaks,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list consisting of one `data.frame` giving the current data. If `data.list` has more than one `data.frame`, only the first element will be used as the current data.
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.

`base.hazard.mean`
: a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.

`base.hazard.sd`
: a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 10.

`get.loglik`
: whether to generate log-likelihood matrix. Defaults to FALSE.

`iter_warmup`
: number of warmup iterations to run per chain. Defaults to 1000. See the argument `iter_warmup` in `sample()` method in cmdstanr package.

`iter_sampling`
: number of post-warmup iterations to run per chain. Defaults to 1000. See the argument `iter_sampling` in `sample()` method in cmdstanr package.

`chains`
: number of Markov chains to run. Defaults to 4. See the argument `chains` in `sample()` method in cmdstanr package.

`...`
: arguments passed to `sample()` method in cmdstanr package (e.g., `seed`, `refresh`, `init`).

Details

The priors on the regression coefficients are independent normal distributions. When the normal priors are elicited with large variances, the prior is also referred to as the reference or vague prior. The baseline hazard parameters are assumed to be independent of the regression coefficients with half-normal priors.

Value

The function returns an object of class `draws_df` containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1690)
    ## take subset for speed purposes
    E1690 = E1690[1:100, ]
    ## replace 0 failure times with 0.50 days
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
```

```
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

---

pwe.pp *Posterior of power prior (PP) with fixed a_0*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard) using the power prior (PP) by Ibrahim and Chen (2000) doi:10.1214/ss/1009212673.

Usage

```
pwe.pp(
  formula,
  data.list,
  breaks,
  a0,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the survival::Surv(time, event) function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of data.frames. The first element in the list is the current data, and the rest are the historical data sets. For fitting piecewise exponential (PWE) models, all historical data sets will be stacked into one historical data set.

breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
a0	a scalar between 0 and 1 giving the (fixed) power prior parameter for the historical data.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, beta.mean will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for beta.mean. Defaults to a vector of 10s.
base.hazard.mean	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 0.
base.hazard.sd	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The power prior parameters (a_0's) are treated as fixed. The initial priors on the regression coefficients are independent normal priors. The current and historical data models are assumed to share the baseline hazard parameters with half-normal priors.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Chen, M.-H. and Ibrahim, J. G. (2000). Power prior distributions for Regression Models. Statistical Science, 15(1).

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      a0 = 0.5,
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`pwe.stratified.pp` *Posterior of stratified power prior (PP) with fixed a_0*

Description

Sample from the posterior distribution of a piecewise exponential (PWE) model (i.e., a proportional hazards model with a piecewise constant baseline hazard) using the power prior (PP) within predefined strata. If the strata and power prior parameters (a_0's) are determined based on propensity scores, this function can be used to sample from the posterior of a PWE model under the propensity score-integrated power prior (PSIPP) by Wang et al. (2019) doi:10.1080/10543406.2019.1657133.

Usage

```
pwe.stratified.pp(
  formula,
  data.list,
```

```
  strata.list,
  breaks,
  a0.strata,
  beta.mean = NULL,
  beta.sd = NULL,
  base.hazard.mean = NULL,
  base.hazard.sd = NULL,
  get.loglik = FALSE,
  iter_warmup = 1000,
  iter_sampling = 1000,
  chains = 4,
  ...
)
```

Arguments

formula	a two-sided formula giving the relationship between the response variable and covariates. The response is a survival object as returned by the `survival::Surv(time, event)` function, where event is a binary indicator for event (0 = no event, 1 = event has occurred). The type of censoring is assumed to be right-censoring.
data.list	a list of `data.frames`. The first element in the list is the current data, and the rest are the historical data sets. For fitting piecewise exponential (PWE) models, all historical data sets will be stacked into one historical data set.
strata.list	a list of vectors specifying the stratum ID for each observation in the corresponding data set in `data.list`. The first element in the list corresponds to the current data, and the rest correspond to the historical data sets. Each vector should have the same length as the number of rows in the respective data set in `data.list`, with values representing stratum labels as positive integers (e.g., 1, 2, 3, ...).
breaks	a numeric vector specifying the time points that define the boundaries of the piecewise intervals. The values should be in ascending order, with the final value being greater than or equal to the maximum observed time.
a0.strata	A scalar or a vector of fixed power prior parameters (a_0's) for each stratum, with values between 0 and 1. If a scalar is provided, it will be replicated for all strata. If a vector is provided, its length must match the total number of unique strata across all data sets. The first element of `a0.strata` corresponds to stratum 1, the second to stratum 2, and so on.
beta.mean	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the mean parameters for the initial prior on regression coefficients. If a scalar is provided, `beta.mean` will be a vector of repeated elements of the given scalar. Defaults to a vector of 0s.
beta.sd	a scalar or a vector whose dimension is equal to the number of regression coefficients giving the sd parameters for the initial prior on regression coefficients. If a scalar is provided, same as for `beta.mean`. Defaults to a vector of 10s.
base.hazard.mean	a scalar or a vector whose dimension is equal to the number of intervals giving the location parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for `beta.mean`. Defaults to 0.

base.hazard.sd	a scalar or a vector whose dimension is equal to the number of intervals giving the scale parameters for the half-normal priors on the baseline hazards of the PWE model. If a scalar is provided, same as for beta.mean. Defaults to 10.
get.loglik	whether to generate log-likelihood matrix. Defaults to FALSE.
iter_warmup	number of warmup iterations to run per chain. Defaults to 1000. See the argument iter_warmup in sample() method in cmdstanr package.
iter_sampling	number of post-warmup iterations to run per chain. Defaults to 1000. See the argument iter_sampling in sample() method in cmdstanr package.
chains	number of Markov chains to run. Defaults to 4. See the argument chains in sample() method in cmdstanr package.
...	arguments passed to sample() method in cmdstanr package (e.g., seed, refresh, init).

Details

The power prior parameters (a_0's) are treated as fixed and must be provided for each stratum. Users must also specify the stratum ID for each observation. Within each stratum, the initial priors on the regression coefficients are independent normal priors, and the current and historical data models are assumed to share the baseline hazard parameters with half-normal priors.

Value

The function returns an object of class draws_df containing posterior samples. The object has two attributes:

data a list of variables specified in the data block of the Stan program

model a character string indicating the model name

References

Wang, C., Li, H., Chen, W.-C., Lu, N., Tiwari, R., Xu, Y., & Yue, L. Q. (2019). Propensity score-integrated power prior approach for incorporating real-world evidence in single-arm clinical studies. Journal of Biopharmaceutical Statistics, 29(5), 731–748.

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## take subset for speed purposes
    E1684 = E1684[1:100, ]
    E1690 = E1690[1:50, ]
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    data_list = list(currdata = E1690, histdata = E1684)
    strata_list = list(rep(1:2, each = 25), rep(1:2, each = 50))
```

```
    # Alternatively, we can determine the strata based on propensity scores
    # using the psrwe package, which is available on GitHub
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    pwe.stratified.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment,
      data.list = data_list,
      strata.list = strata_list,
      breaks = breaks,
      a0.strata = c(0.3, 0.5),
      chains = 1, iter_warmup = 500, iter_sampling = 1000
    )
  }
}
```

`sample.ensemble` *Sample from the ensemble posterior distribution*

Description

Given model averaging weights (e.g., from Bayesian model averaging (BMA), pseudo-BMA, or stacking) and a matrix of posterior samples from the candidate models, this function draws samples from the model-averaged posterior distribution. Here, each "model" refers to a unique combination of an outcome model and its associated priors. Posterior draws are randomly selected from the candidate models in proportion to their specified weights, producing samples from the ensemble of posterior distributions.

Usage

```
sample.ensemble(wts, samples.mtx)
```

Arguments

`wts`	a numeric vector of normalized model averaging weights (e.g., from `compute.ensemble.weights()`). The length of `wts` must match the number of columns in `samples.mtx`.
`samples.mtx`	a matrix of posterior samples. Each column corresponds to samples from a different model, and each row is one posterior draw (e.g., from Markov chain Monte Carlo (MCMC) sampling). All columns must have the same number of samples.

Details

This function is typically used in combination with compute.ensemble.weights(), which computes model averaging weights using methods such as Bayesian model averaging (BMA), pseudo-BMA, pseudo-BMA with the Bayesian bootstrap, or stacking). The input matrix of posterior samples should have one column per candidate model, with each column containing posterior draws from that model.

Value

The function returns a numeric vector of ensemble posterior draws, sampled proportionally to the provided model weights. The returned vector has the same length as the number of rows in samples.mtx.

See Also

compute.ensemble.weights()

Examples

```
if (instantiate::stan_cmdstan_exists()) {
  if(requireNamespace("survival")){
    library(survival)
    data(E1684)
    data(E1690)
    ## replace 0 failure times with 0.50 days
    E1684$failtime[E1684$failtime == 0] = 0.50/365.25
    E1690$failtime[E1690$failtime == 0] = 0.50/365.25
    E1684$cage = as.numeric(scale(E1684$age))
    E1690$cage = as.numeric(scale(E1690$age))
    data_list = list(currdata = E1690, histdata = E1684)
    nbreaks = 3
    probs   = 1:nbreaks / nbreaks
    breaks  = as.numeric(
      quantile(E1690[E1690$failcens==1, ]$failtime, probs = probs)
    )
    breaks  = c(0, breaks)
    breaks[length(breaks)] = max(10000, 1000 * breaks[length(breaks)])
    fit.pwe.pp = pwe.pp(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      a0 = 0.5,
      get.loglik = TRUE,
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
    fit.pwe.post = pwe.post(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      get.loglik = TRUE,
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
```

```
    )
    fit.pwe.commensurate = pwe.commensurate(
      formula = survival::Surv(failtime, failcens) ~ treatment + sex + cage + node_bin,
      data.list = data_list,
      breaks = breaks,
      p.spike = 0.1,
      get.loglik = TRUE,
      chains = 1, iter_warmup = 1000, iter_sampling = 2000
    )
    fit.list = list(fit.pwe.post, fit.pwe.pp, fit.pwe.commensurate)
    samples.mtx = do.call(
     cbind, lapply(fit.list, function(d){
       as.numeric( d[["treatment"]] )
     })
    )
    wts = compute.ensemble.weights(
      fit.list = fit.list,
      type = "pseudobma+"
    )$weights
    sample.ensemble(
     wts = wts, samples.mtx = samples.mtx
    )
  }
}
```

Index