Package 'shrinkDSM'

July 23, 2025

Type Package

Title Efficient Bayesian Inference for Dynamic Survival Models with Shrinkage

Version 1.0.0

Description Efficient Markov chain Monte Carlo (MCMC) algorithms for fully Bayesian estimation of dynamic survival models with shrinkage priors. Details on the algorithms used are provided in Wagner (2011) <doi:10.1007/s11222-009-9164-5>, Bitto and Frühwirth-Schnatter (2019) <doi:10.1016/j.jeconom.2018.11.006> and Cadonna et al. (2020) <doi:10.3390/econometrics8020020>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0),

Imports Rcpp, stochvol (>= 3.0.3), coda, utils, shrinkTVP (>= 2.0.2), survival

LinkingTo Rcpp, RcppArmadillo, RcppProgress, stochvol, shrinkTVP

RoxygenNote 7.3.2

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Daniel Winkler [aut, cre], Peter Knaus [aut] (ORCID: <https://orcid.org/0000-0001-6498-7084>)

Maintainer Daniel Winkler <daniel.winkler@wu.ac.at>

Repository CRAN

Date/Publication 2025-04-11 22:00:02 UTC

Contents

divisionpoints 2
gastric 3
plot.mcmc.dsm.tvp 4
plot.shrinkDSM 6
plot.shrinkDSM_pred 8
predict.shrinkDSM 9
prep_tvinput 10
print.shrinkDSM 12
shrinkDSM 13

Index **20**

`divisionpoints` *Create division points for estimation of a dynamic survival model*

Description

Create a vector of division points for the model. These points mark the times at which the parameters are allowed to evolve, with the parameters being fixed between division points. The points are generated in a data driven fashion, with a new point being created when `events` number of interesting events have been observed since the last division point.

Usage

```
divisionpoints(times, delta, events = 1)
```

Arguments

`times`	A vector of real, positive numbers indicating the survival times. For right censored data, this is the follow up time.
`delta`	A vector of status indicators, with 0 = alive and 1 = dead. Other choices are `TRUE`/`FALSE` (`TRUE` = death) or 1/2 (2 = death).
`events`	A positive integer indicating the number of interesting events per interval until a new division is created.

Value

Returns an integer vector of time points to be used as division points `S` in `shrinkDSM`.

Author(s)

Daniel Winkler <daniel.winkler@wu.ac.at>

Examples

```
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)
```

---

gastric *Survival times of gastric cancer patients*

Description

A data set of survival times of patients with locally advanced, nonresectable gastric carcinoma. The patients were either treated with chemotherapy plus radiation or chemotherapy alone.

Usage

```
gastric
```

Format

A data frame with 90 rows and 4 variables:

id patient id

radiation dummy variable indicating which treatment was employed, 0 = chemotherapy, 1 = combined chemotherapy/radiation

time time survived by patient in days

status dummy variable indicating whether death of the patient was observed, 0 = death not observed (i.e. censored), 1 = death observed.

Source

Moreau, T., O'Quigley, J., and Mesbah M. (1985) A global goodness-of-fit statistic for the proportional hazards model Appl. Statist., 34, 212:218 (p 213)

https://www.mayo.edu/research/documents/gastrichtml/DOC-10027680

`plot.mcmc.dsm.tvp` *Graphical summary of posterior distribution for a piecewise constant, time-varying parameter*

Description

`plot.mcmc.dsm.tvp` plots empirical posterior quantiles for a piecewise constant, time-varying parameter.

Usage

```
## S3 method for class 'mcmc.dsm.tvp'
plot(
  x,
  probs = c(0.025, 0.25, 0.75, 0.975),
  shaded = TRUE,
  quantlines = FALSE,
  shadecol = "skyblue",
  shadealpha = 0.5,
  quantlty = 2,
  quantcol = "black",
  quantlwd = 0.5,
  drawzero = TRUE,
  zerolty = 2,
  zerolwd = 1,
  zerocol = "grey",
  ...
)
```

Arguments

`x`	`mcmc.dsm.tvp` object
`probs`	vector of boundaries for credible intervals to plot for each point in time, with values in [0,1]. The largest and smallest value form the outermost credible interval, the second smallest and second largest the second outermost and so forth. The default value is `c(0.025, 0.25, 0.75, 0.975)`. Note that there have to be the same number of probs < 0.5 as there are > 0.5.
`shaded`	single logical value or a vector of logical values, indicating whether or not to shade the area between the pointwise credible intervals. If a vector is given, the first value given is used to determine if the area between the outermost credible interval is shaded, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of quantile pairs. The default value is `TRUE`.
`quantlines`	single logical value or a vector of logical values, indicating whether or not to draw borders along the pointwise credible intervals. If a vector is given, the first value given is used to determine whether the outermost credible interval is

marked by lines, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of credible intervals. The default value is FALSE.

shadecol — single character string or a vector of character strings. Determines the color of the shaded areas that represent the credible intervals. If a vector is given, the first color given is used for the outermost area, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of shaded areas. Has no effect if shaded = FALSE. The default value is "skyblue".

shadealpha — real number between 0 and 1 or a vector of real numbers between 0 and 1. Determines the level of transparency of the shaded areas that represent the credible intervals. If a vector is given, the first value given is used for the outermost area, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of shaded areas. Has no effect if shaded = FALSE. The default value is 0.5.

quantlty — either a single integer in [0,6] or one of the character strings "blank","solid", "dashed", "dotted", "dotdash", "longdash", "twodash" or a vector containing these. Determines the line type of the borders drawn around the shaded areas that represent the credible intervals. Note that if a vector is supplied the elements have to either be all integers or all character strings. If a vector is given, the first value given is used for the outermost area, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of shaded areas. Has no effect if quantlines = FALSE. The default value is 2.

quantcol — single character string or a vector of character strings. Determines the color of the borders drawn around the shaded areas that represent the credible intervals. If a vector is given, the first color given is used for borders of the outermost area, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of shaded areas. Has no effect if quantlines = FALSE. The default value is "black".

quantlwd — single real, positive number or a vector of real, positive numbers. Determines the line width of the borders drawn around the shaded areas that represent the credible intervals. If a vector is given, the first number given is used for the borders of the outermost area, the second for the second outermost and so forth. Recycled in the usual fashion if the vector is shorter than the number of shaded areas. Has no effect if quantlines = FALSE. The default value is 0.5.

drawzero — single logical value determining whether to draw a horizontal line at zero or not. The default value is TRUE.

zerolty — single integer in [0,6] or one of the character strings "blank","solid", "dashed", "dotted", "dotdash", "longdash", "twodash". Determines the line type of the horizontal line at zero. Has no effect if drawzero = FALSE. The default value is 2.

zerolwd — single real, positive number. Determines the line width of the horizontal line at zero. Has no effect if drawzero = FALSE. The default value is 1.

zerocol — single character string. Determines the color of the horizontal line at zero. Has no effect if drawzero = FALSE. The default value is "grey".

... — further arguments to be passed to plot.

Value

Called for its side effects and returns invisibly.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

Other plotting functions: plot.shrinkDSM(), plot.shrinkDSM_pred()

Examples

```
set.seed(123)
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)

# Estimate model
mod <- shrinkDSM(time ~ radiation, gastric,
                 delta = gastric$status, S = intervals)

# Plot piecewise constant, time-varying parameter
plot(mod$beta$beta_radiation)
```

---

plot.shrinkDSM *Graphical summary of posterior distribution of fitted dynamic survival model*

Description

plot.shrinkDSM generates plots visualizing the posterior distribution estimated as a result from a call to shrinkDSM.

Usage

```
## S3 method for class 'shrinkDSM'
plot(
  x,
  pars = c("beta"),
  nplot = 3,
  h_borders = c(0.05, 0.05),
  w_borders = c(0.02, 0.02),
  ...
)
```

Arguments

x	a shrinkDSM object.
pars	a character vector containing the names of the parameters to be visualized. The names have to coincide with the names of the list elements of the shrinkDSM object. Throws an error if any element of pars does not fulfill this criterium. The default is c("beta").
nplot	positive integer that indicates the number of tvp plots to display on a single page before a new page is generated. The default value is 3.
h_borders	single real, positive number smaller than 0.5 or a vector containing two such numbers. Determines the relative amount of space (the total amount summing up to 1) left blank on the left and right of the plot, in that order. The default is c(0.05, 0.05).
w_borders	single real, positive number smaller than 0.5 or a vector containing two such numbers. Determines the relative amount of space (the total amount summing up to 1) left blank at the top and bottom of the plot, in that order. The default is c(0.02, 0.02).
...	further arguments to be passed to the respective plotting functions.

Value

Called for its side effects and returns invisibly.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

See Also

Other plotting functions: plot.mcmc.dsm.tvp(), plot.shrinkDSM_pred()

Examples

```
set.seed(123)
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)

# Estimate model
mod <- shrinkDSM(time ~ radiation, gastric,
                 delta = gastric$status, S = intervals)
plot(mod)


# Will produce an error because 'hello' is not a parameter in the model
## Not run:
plot(mod, pars = c("beta", "hello"))

## End(Not run)
```

plot.shrinkDSM_pred *Graphical summary of posterior predictive density*

Description

plot.shrinkDSM_pred generates plots visualizing the posterior predictive density generated by predict.shrinkDSM.

Usage

```
## S3 method for class 'shrinkDSM_pred'
plot(x, dens_args, legend = TRUE, ...)
```

Arguments

x	a shrinkDSM_pred object.
dens_args	*optional* named list containing arguments passed to density.
legend	logical value inidicating whether a legend should be added. Defaults to TRUE.
...	further arguments to be passed to plots.

Value

Called for its side effects and returns invisibly.

See Also

Other plotting functions: plot.mcmc.dsm.tvp(), plot.shrinkDSM()

Examples

```
set.seed(123)
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)

# Estimate model
mod <- shrinkDSM(time ~ radiation, gastric,
                 delta = gastric$status, S = intervals)

# Draw from posterior predictive distribution
newdata <- data.frame(radiation = c(0, 1))
pred <- predict(mod, newdata = newdata)

# Plot predictions
plot(pred)
```

predict.shrinkDSM *Draw from posterior predictive density of a fitted time-varying parameter survival model*

Description

Draws from the posterior predictive distribution of survival times based on a fitted time-varying parameter survival model resulting from a call to shrinkDSM.

Usage

```
## S3 method for class 'shrinkDSM'
predict(object, newdata, cens = TRUE, ...)
```

Arguments

object	an object of class shrinkDSM, containing the fitted model.
newdata	a data frame containing the covariates used for the prediction. The names of the covariates have to match the names used during model estimation in the call to shrinkDSM.
cens	logical value indicating whether the predictions should be censored at the largest survival time in the data used for estimation. The default value is TRUE.
...	included for S3 method consistency and currently ignored.

Value

The value returned is a list object of class shrinkTVP_pred containing the samples from the posterior predictive density.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

Daniel Winkler <dwinkler@wu.ac.at>

Examples

```
set.seed(123)
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)

# Estimate model
mod <- shrinkDSM(time ~ radiation, gastric,
                 delta = gastric$status, S = intervals)

# Draw from posterior predictive distribution
```

```
newdata <- data.frame(radiation = c(0, 1))
pred <- predict(mod, newdata = newdata)
```

`prep_tvinput` *Prepare time-varying inputs for estimation of a dynamic survival model*

Description

This function pre-processes time-varying inputs in such a way that `shrinkDSM` can work with time-varying inputs. Its main inputs are two data frames, namely `surv_data` and `covariate_data`. `surv_data` contains meta data about each observation (i.e. survival time and censoring indicator), while `covariate_data` contains the time-varying covariates (one per observation and time interval) and an index for which time interval each covariate is observed in. The two are merged together via an ID that needs to be unique for each observation and present in both `surv_data` and `covariate_data`.

Usage

```
prep_tvinput(
  surv_data,
  covariate_data,
  id_var,
  surv_var,
  delta_var,
  interval_var,
  covariate_id_var = id_var
)
```

Arguments

`surv_data`	data frame containing meta-data for each observation (survival time and censoring indicator) as well as an ID for each observation.
`covariate_data`	data frame containing the time-varying covariates (one per observation and time interval), an index for which time interval each covariate is observed in as well as an ID for each observation.
`id_var`	character string specifying the column name of the ID variable. If the name is different in `surv_data` and `covariate_data`, `id_var` will be used in `surv_data`, whereas `covariate_id_var` will be used in `covariate_data`.
`surv_var`	character string specifying the column name of the survival times in `surv_data`.
`delta_var`	character string specifying the column name of the status indicators in `surv_data`, 0 = censored or 1 = event observed..
`interval_var`	character string specifying the column name of the time interval ID in `covariate_data`.
`covariate_id_var`	character string specifying the column name of the ID variable in `covariate_data`. Defaults to `id_var`.

Value

Returns an object of class `data.frame` and `tvsurv` to be used as an input in `shrinkDSM`.

Author(s)

Daniel Winkler <daniel.winkler@wu.ac.at>

Peter Knaus <peter.knaus@wu.ac.at>

Examples

```
# A toy example with 5 observations and 2 covariates, observed over 3 time periods
set.seed(123)
n_obs <- 5
surv_var <- round(rgamma(n_obs, 1, .1)) + 1
delta_var <- sample(size = n_obs, c(0, 1), prob = c(0.2, 0.8), replace = TRUE)

surv_data <- data.frame(id_var = 1:n_obs, surv_var, delta_var)

# Determine intervals
S <- c(3, 11)

# Create synthetic observations for each individual
covariate_list <- list()

for (i in 1:n_obs) {
  nr_periods_survived <- sum(surv_var[i] > S) + 1
  covariate_list[[i]] <- data.frame(id_var = i,
                                    interval_var = 1:nr_periods_survived,
                                    x1 = rnorm(nr_periods_survived),
                                    x2 = rnorm(nr_periods_survived))
}

# Bind all individual covariate data frames together
# Each observation now has a covariate in each period they
# were observed in.
covariate_data <- do.call(rbind, covariate_list)

# Call prep_tvinput to pre-process for shrinkDSM
merged_data <- prep_tvinput(surv_data,
                            covariate_data,
                            id_var = "id_var",
                            surv_var = "surv_var",
                            delta_var = "delta_var",
                            interval_var = "interval_var")

# Can now be used in shrinkDSM
# Note that delta is now automatically extracted from merged_data,
# providing it will throw a warning
mod <- shrinkDSM(surv_var ~ x1 + x2, merged_data, S = S)
```

`print.shrinkDSM` *Nicer printing of shrinkDSM objects*

Description

Nicer printing of shrinkDSM objects

Usage

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

Arguments

`x`	a `shrinkDSM` object.
`...`	Currently ignored.

Value

Called for its side effects and returns invisibly.

Author(s)

Peter Knaus <peter.knaus@wu.ac.at>

Examples

```
set.seed(123)
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)

# Estimate model
mod <- shrinkDSM(time ~ radiation, gastric,
                 delta = gastric$status, S = intervals)

# Print
print(mod)

# Alternatively
mod
```

shrinkDSM *Markov Chain Monte Carlo (MCMC) for time-varying parameter survival models with shrinkage*

Description

shrinkDSM samples from the joint posterior distribution of the parameters of a time-varying parameter survival model with shrinkage and returns the MCMC draws. See also shrinkTVP to see more examples of how to modify the prior setup of the time-varying component of the model.

Usage

```
shrinkDSM(
  formula,
  data,
  mod_type = "double",
  delta,
  S,
  group,
  subset,
  niter = 10000,
  nburn = round(niter/2),
  nthin = 1,
  learn_a_xi = TRUE,
  learn_a_tau = TRUE,
  a_xi = 0.1,
  a_tau = 0.1,
  learn_c_xi = TRUE,
  learn_c_tau = TRUE,
  c_xi = 0.1,
  c_tau = 0.1,
  a_eq_c_xi = FALSE,
  a_eq_c_tau = FALSE,
  learn_kappa2_B = TRUE,
  learn_lambda2_B = TRUE,
  kappa2_B = 20,
  lambda2_B = 20,
  hyperprior_param,
  sv_param,
  MH_tuning,
  phi_param,
  display_progress = TRUE
)
```

Arguments

formula — an object of class "formula": a symbolic representation of the model, as in the function lm. For details, see formula. The response may by a survival object as

returned by the Surv function.

data — *optional* data frame containing the response variable and the covariates. If not found in data, the variables are taken from environment(formula), typically the environment from which shrinkDSM is called. No NAs are allowed in the response variable and the covariates.

mod_type — character string that reads either "triple", "double" or "ridge". Determines whether the triple gamma, double gamma or ridge prior are used for theta_sr and beta_mean. The default is "double".

delta — The status indicator of the last period, 0 = censored or 1 = event observed. If the formula is a survival object this parameter is ignored.

S — integer vector of time points that start a new interval. Parameters are fixed within an interval and vary across intervals.

group — *optional* grouping indicator for latent factor.

subset — *optional* vector specifying a subset of observations to be used in the fitting process.

niter — positive integer, indicating the number of MCMC iterations to perform, including the burn-in. Has to be larger than or equal to nburn + 2. The default value is 10000.

nburn — non-negative integer, indicating the number of iterations discarded as burn-in. Has to be smaller than or equal to niter - 2. The default value is round(niter / 2).

nthin — positive integer, indicating the degree of thinning to be performed. Every nthin draw is kept and returned. The default value is 1, implying that every draw is kept.

learn_a_xi — logical value indicating whether to learn a_xi, the spike parameter of the state variances. The default value is TRUE.

learn_a_tau — logical value indicating whether to learn a_tau, the spike parameter of the mean of the initial values of the states. The default value is TRUE.

a_xi — positive, real number, indicating the (fixed) value for a_xi. Ignored if learn_a_xi is TRUE or mod_type is set to "ridge". The default value is 0.1.

a_tau — positive, real number, indicating the (fixed) value for a_tau. Ignored if learn_a_tau is TRUE or mod_type is set to "ridge". The default value is 0.1.

learn_c_xi — logical value indicating whether to learn c_xi, the tail parameter of the state variances. Ignored if mod_type is not set to "triple" or a_eq_c_xi is set to TRUE. The default value is TRUE.

learn_c_tau — logical value indicating whether to learn c_tau, the tail parameter of the mean of the initial values of the states. Ignored if mod_type is not set to "triple" or a_eq_c_tau is set to TRUE. The default value is TRUE.

c_xi — positive, real number, indicating the (fixed) value for c_xi. Ignored if learn_c_xi is TRUE, mod_type is not set to "triple" or a_eq_c_xi is set to TRUE. The default value is 0.1.

c_tau — positive, real number, indicating the (fixed) value for c_tau. Ignored if learn_c_xi is TRUE, mod_type is not set to "triple" or a_eq_c_tau is set to TRUE. The default value is 0.1.

a_eq_c_xi logical value indicating whether to force a_xi and c_xi to be equal. If set to TRUE, beta_a_xi and alpha_a_xi are used as the hyperparameters and beta_c_xi and alpha_c_xi are ignored. Ignored if mod_type is not set to "triple". The default value is FALSE.

a_eq_c_tau logical value indicating whether to force a_tau and c_tau to be equal. If set to TRUE, beta_a_tau and alpha_a_tau are used as the hyperparameters and beta_c_tau and alpha_c_tau are ignored. Ignored if mod_type is not set to "triple". The default value is FALSE.

learn_kappa2_B logical value indicating whether to learn kappa2_B, the global level of shrinkage for the state variances. The default value is TRUE.

learn_lambda2_B
logical value indicating whether to learn the lambda2_B parameter, the global level of shrinkage for the mean of the initial values of the states. The default value is TRUE.

kappa2_B positive, real number. Initial value of kappa2_B. The default value is 20.

lambda2_B positive, real number. Initial value of lambda2_B. The default value is

hyperprior_param
optional named list containing hyperparameter values. Not all have to be supplied, with those missing being replaced by the default values. Any list elements that are misnamed will be ignored and a warning will be thrown. All hyperparameter values have to be positive, real numbers. The following hyperparameters can be supplied:

- e1: The default value is 0.001.
- e2: The default value is 0.001.
- d1: The default value is 0.001.
- d2: The default value is 0.001.
- beta_a_xi: The default value is 10.
- beta_a_tau: The default value is 10.
- alpha_a_xi: The default value is 5.
- alpha_a_tau: The default value is 5.

sv_param *optional* named list containing hyperparameter values for the stochastic volatility parameters. Not all have to be supplied, with those missing being replaced by the default values. Any list elements that are misnamed will be ignored and a warning will be thrown. Ignored if group is missing. The following elements can be supplied:

- Bsigma_sv: positive, real number. The default value is 1.
- a0_sv: positive, real number. The default value is 5.
- b0_sv: positive, real number. The default value is 1.5.

MH_tuning *optional* named list containing values used to tune the MH steps for a_xi and a_tau. Not all have to be supplied, with those missing being replaced by the default values. Any list elements that are misnamed will be ignored and a warning will be thrown. The arguments for a_xi(a_tau) are only used if learn_a_xi(learn_a_tau) is set to TRUE. Arguments ending in "adaptive" are logical values indicating whether or not to make the MH step for the respective parameter adaptive. Arguments ending in "tuning_par" serve two different

purposes. If the respective MH step is not set to be adaptive, it acts as the standard deviation of the proposal distribution. If the respective MH step is set to be adaptive, it acts as the initial standard deviation. Arguments ending in "target_rate" define the acceptance rate the algorithm aims to achieve. Arguments ending in "max_adapt" set the maximum value by which the logarithm of the standard deviation of the proposal distribution is adjusted. Finally, arguments ending in "batch_size" set the batch size after which the standard deviation of the proposal distribution is adjusted. The following elements can be supplied:

- `a_xi_adaptive`: logical value. The default is `TRUE`.
- `a_xi_tuning_par`: positive, real number. The default value is 1.
- `a_xi_target_rate`: positive, real number, between 0 and 1. The default value is 0.44.
- `a_xi_max_adapt`: positive, real number. The default value is 0.01.
- `a_xi_batch_size`: positive integer. The default value is 50.
- `a_tau_adaptive`: logical value. The default is `TRUE`.
- `a_tau_tuning_par`: positive, real number. The default value is 1.
- `a_tau_target_rate`: positive, real number, between 0 and 1. The default value is 0.44.
- `a_tau_max_adapt`: positive, real number. The default value is 0.01.
- `a_tau_batch_size`: positive integer. The default value is 50.

`phi_param` *optional* named list containing hyperparameter values for the grouped factor and values to tune the MH steps for `a_phi` and `c_phi`. Not all have to be supplied, with those missing being replaced by the default values. Any list elements that are misnamed will be ignored and a warning will be thrown. Ignored if `group` is missing. The following elements can be supplied:

- `mod_type_phi` character string that reads either `"triple"`, `"double"` or `"ridge"`. Determines whether the triple gamma, double gamma or ridge prior are used for `phi`. The default is "double".
- `learn_a_phi`: logical value. The default is `TRUE`.
- `a_phi`: positive, real number. The default value is 0.1.
- `learn_c_phi`: logical value. The default is `TRUE`.
- `c_phi`: positive, real number. The default value is 0.1,
- `a_phi_eq_c_phi`: logical value. The default is `FALSE`.
- `learn_lambda2_B_phi`: logical value. The default is `TRUE`.
- `lambda2_B_phi`: positive, real number. The default value is 20.
- `e1_phi`: positive, real number. The default value is 0.001.
- `e2_phi`: positive, real number. The default value is 0.001.
- `beta_a_phi`: positive, real number. The default value is 10.
- `alpha_a_phi`: positive, real number. The default value is 5.
- `beta_c_phi`: positive, real number. The default value is 10.
- `alpha_c_phi`: positive, real number. The default value is 5.
- `a_phi_adaptive`: logical value. The default is `TRUE`.
- `a_phi_tuning_par`: positive, real number. The default value is 1.

- a_phi_target_rate: positive, real number, between 0 and 1. The default value is 0.44.
- a_phi_max_adapt: positive, real number. The default value is 0.01.
- a_phi_batch_size: positive integer. The default value is 50.
- c_phi_adaptive: logical value. The default is TRUE.
- c_phi_tuning_par: positive, real number. The default value is 1.
- c_phi_target_rate: positive, real number, between 0 and 1. The default value is 0.44.
- c_phi_max_adapt: positive, real number. The default value is 0.01.
- c_phi_batch_size: positive integer. The default value is 50.

display_progress

logical value indicating whether the progress bar and other informative output should be displayed. The default value is TRUE.

Value

The value returned is a list object of class shrinkDSM containing

beta — list object containing an mcmc.dsm.tvp object for the parameter draws from the posterior distribution of the centered states, one for each covariate. In the case that there is only one covariate, this becomes just a single mcmc.dsm.tvp object.

beta_mean — mcmc object containing the parameter draws from the posterior distribution of beta_mean.

theta_sr — mcmc object containing the parameter draws from the posterior distribution of the square root of theta.

tau2 — mcmc object containing the parameter draws from the posterior distribution of tau2.

xi2 — mcmc object containing the parameter draws from the posterior distribution of xi2.

lambda2 — *(optional)* mcmc object containing the parameter draws from the posterior distribution of lambda2. Not returned if mod_type is not "triple".

kappa2 — *(optional)* mcmc object containing the parameter draws from the posterior distribution of kappa2. Not returned if mod_type is not "triple".

a_xi — *(optional)* mcmc object containing the parameter draws from the posterior distribution of a_xi. Not returned if learn_a_xi is FALSE or mod_type is "ridge".

a_tau — *(optional)* mcmc object containing the parameter draws from the posterior distribution of a_tau. Not returned if learn_a_tau is FALSE or mod_type is "ridge".

c_xi — *(optional)* mcmc object containing the parameter draws from the posterior distribution of c_xi. Not returned if learn_c_xi is FALSE or mod_type is not "triple".

c_tau — *(optional)* mcmc object containing the parameter draws from the posterior distribution of c_tau. Not returned if learn_c_tau is FALSE or mod_type is not "triple".

lambda2_B	*(optional)* mcmc object containing the parameter draws from the posterior distribution of lambda2_B. Not returned if learn_lambda2_B is FALSE or mod_type is "ridge".
kappa2_B	*(optional)* mcmc object containing the parameter draws from the posterior distribution of kappa2_B. Not returned if learn_kappa2_B is FALSE or mod_type is "ridge".
MH_diag	*(optional)* named list containing statistics for assessing MH performance. Not returned if no MH steps are required or none of them are specified to be adaptive.
priorvals	list object containing hyperparameter values of the prior distributions, as specified by the user.
model	list object containing the model matrix, model response and formula used.
summaries	list object containing a collection of summary statistics of the posterior draws.

To display the output, use plot and summary. The summary method displays the specified prior values stored in priorvals and the posterior summaries stored in summaries, while the plot method calls coda's plot.mcmc or the plot.mcmc.dsm.tvp method. Furthermore, all functions that can be applied to coda::mcmc objects (e.g. coda::acfplot) can be applied to all output elements that are coda compatible.

Author(s)

Daniel Winkler <daniel.winkler@wu.ac.at>

Peter Knaus <peter.knaus@wu.ac.at>

Examples

```
set.seed(123)
data("gastric")

# Create intervals for piecewise exponential model
intervals <- divisionpoints(gastric$time, gastric$status, 2)

# Estimate baseline model
mod <- shrinkDSM(time ~ radiation, gastric,
  delta = gastric$status, S = intervals
)

# Alternative formula interface
mod_surv <- shrinkDSM(Surv(time, status) ~ radiation, gastric,
  S = intervals
)

# Estimate model with different prior setup
mod2 <- shrinkDSM(time ~ radiation, gastric,
  delta = gastric$status, S = intervals,
  mod_type = "triple"
)

# Change some of the hyperparameters
```

```
mod3 <- shrinkDSM(time ~ radiation, gastric,
  delta = gastric$status, S = intervals,
  mod_type = "triple",
  hyperprior_param = list(
    beta_a_xi = 5,
    alpha_a_xi = 10
  )
)
```

Index

∗ **datasets**
gastric, 3
∗ **plotting functions**
plot.mcmc.dsm.tvp, 4
plot.shrinkDSM, 6
plot.shrinkDSM_pred, 8
∗ **prediction functions**
predict.shrinkDSM, 9

divisionpoints, 2

formula, *13*

gastric, 3

plot.mcmc.dsm.tvp, 4, *7, 8*
plot.shrinkDSM, *6*, 6, *8*
plot.shrinkDSM_pred, *6, 7*, 8
predict.shrinkDSM, 9
prep_tvinput, 10
print.shrinkDSM, 12

shrinkDSM, 13
shrinkTVP, *13*