Package 'HPR'

April 12, 2022

Type Package

Title Horseshoe Process Regression

Version 0.1.0

Description HPR performs horseshoe process regression, as described in the article by Chase et al. (2022+). HPR is an analogue to Gaussian process regression (GPR), but intended for more abruptly-varying functions, like step functions, than can be modeled by GPR. It allows for normal, Poisson, and binomial outcome data, with a mixture of HPR and linear predictors, which can be constrained to be monotonic increasing or decreasing. For more information, see Chase et al. (2022+) and the vignette.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

Imports dplyr,
cmdstanr,
posterior,
mgcv,
stats,
knitr,
Rdpack

VignetteBuilder knitr

RdMacros Rdpack

RoxygenNote 7.1.2

Suggests rmarkdown

R topics documented:

get_betas *Function to extract the estimates of the linear coefficients from an HPR model*

Description

Function to extract the estimates of the linear coefficients from an HPR model

Usage

```
get_betas(object = NULL, alpha = 0.05, var_names = NULL)
```

Arguments

object	The results object from a run of hpr for which a matrix Z was input.
alpha	The uncertainty level for the betas; the default is 0.05 (which corresponds to 95% credible intervals).
var_names	An optional character vector containing the variable names of the columns of Z.

Value

A dataframe with a row for each beta and columns:

Variable the name of the variable, if var_names was used

Mean the mean of the posterior samples of the beta

Median the median of the posterior samples of the beta

Lower the alpha/2 percentile of the posterior samples of the beta

Upper the 1-alpha/2 percentile of the posterior samples of the beta

Examples

```
X <- as.matrix(dat$Day, ncol = 1)
y <- dat$Temperature
Z <- as.matrix(dat$Aberration, ncol = 1)

mymodel <- hpr(y = y, X = X, Z = Z, family = "gaussian", beta_dist = "cauchy")
mybetas <- get_betas(mymodel, var_names = c("Aberration"))
```

`get_diagnostics` *Function to extract Stan diagnostics from an HPR model*

Description

Function to extract Stan diagnostics from an HPR model

Usage

```
get_diagnostics(object = NULL, verbose = FALSE)
```

Arguments

`object`	The results object from a run of hpr.
`verbose`	A logical indicator of whether a full cmdstan diagnostic report should be printed to the console. The default is false.

Value

A dataframe with columns:

Divergences The number of HMC samples that ended in a divergence.

Max_Treedepth The number of HMC samples that had a max_treedepth warning.

Rhat The number of f parameters that had Rhat greater than 1.1, using the adjusted Rhat of Vehtari et al. (Bayesian Analysis, 2021).

Min_Ess_Bulk The minimum effective sample size in the bulk of the posterior across the f parameters. This estimated according to Vehtari et al. (Bayesian Analysis, 2021).

Min_Ess_Tail The minimum effective sample size in the tails of the posterior across the f parameters. This estimated according to Vehtari et al. (Bayesian Analysis, 2021).

Num_Param The length of the f vector (the systematic component of the model), which is a function of all other parameters in the model.

Num_Samples The number of HMC posterior samples.

Time The computing time of Stan sampling.

For more information on these metrics, please see Chase et al. (2022+) or the Stan reference manual.

Examples

```
X <- as.matrix(dat$Day, ncol = 1)
y <- dat$Temperature

mymodel <- hpr(y = y, X = X, family = "gaussian")
get_diagnostics(mymodel)
```

get_f *Function to extract the nonlinear component of a horseshoe model*

Description

Function to extract only the nonlinear component of a predicted horseshoe curve fit from an HPR model at each unique, ordered value of X. If no Z matrix was used, get_f will return the same results as get_preds. If a Z matrix was used, get_f will return the nonlinear component estimated at the average value of each predictor in Z.

Usage

```
get_f(object = NULL, alpha = 0.05)
```

Arguments

object	The results object from a run of hpr.
alpha	The uncertainty level for the nonlinear component of f; the default is 0.05 (which corresponds to 95% credible intervals).

Value

A dataframe with m x q rows (corresponding to the unique, ordered values of each column of X) and columns:

x The value of X

Median the median of the posterior samples of the nonlinear component

Mean the mean of the posterior samples of the nonlinear component

Lower the alpha/2 percentile of the posterior samples of the nonlinear component

Upper the 1-alpha/2 percentile of the posterior samples of the nonlinear component

Predictor if q > 1, then this gives the column index of X for which the nonlinear component is being estimated

Examples

```
X <- as.matrix(dat$Day, ncol = 1)
y <- dat$Temperature

mymodel <- hpr(y = y, X = X, family = "gaussian")
my_nonlin <- get_f(mymodel)
```

get_preds *Function to extract f, the systematic component of an HPR model*

Description

Function to extract the estimated value of f (the systematic component of the model) from an HPR model. For posterior predictions, use post_predict.

Usage

```
get_preds(object = NULL, alpha = 0.05)
```

Arguments

object	The results object from a run of hpr.
alpha	The uncertainty level for the values of f; the default is 0.05 (which corresponds to 95% credible intervals).

Value

A dataframe with N rows (corresponding to the rows of X, Z) and columns:

Mean the mean of the posterior samples of f

Median the median of the posterior samples of f

Lower the alpha/2 percentile of the posterior samples of f

Upper the 1-alpha/2 percentile of the posterior samples of f

Examples

```
X <- as.matrix(dat$Day, ncol = 1)
y <- dat$Temperature

mymodel <- hpr(y = y, X = X, family = "gaussian")
my_f <- get_preds(mymodel)
my_f$x <- dat$Day
```

hpr *Function to fit a Bayesian horseshoe process regression*

Description

This function fits a Bayesian horseshoe process regression: it calculates the posterior distribution for a set of q horseshoe process terms on some continuous predictors X, and a set of p linear predictors Z. Outcomes can be Gaussian, binary, or Poisson distributed. Monotonic constraints can be enforced on the horseshoe process terms, if desired. At present, this model is implemented for q = 1 or 2, although future versions may permit higher values for q. If the predictor(s) X are sparsely distributed over their domain, it may be wise to add some additional augmented data to even out the grid of X or to obtain predictions/interpolations where desired. More information can be found in Chase et al. (2022+).

Usage

```
hpr(
  y = NULL,
  X = NULL,
  Z = NULL,
  X_aug = NULL,
  family = "gaussian",
  beta_dist = "normal",
  beta_mean = NULL,
  beta_sd = NULL,
  beta_df = NULL,
  monotonic_terms = NULL,
  monotonic_approach = NULL,
  aug_approach = "MCMC",
  new_X = NULL,
  new_Z = NULL,
  seed = NULL,
  chains = 4,
  iter_warmup = 1000,
  iter_sampling = 2000,
  max_treedepth = 12,
  adapt_delta = 0.95,
  c = 0.01,
  verbose = FALSE
)
```

Arguments

y	A length N numeric vector containing the outcome of interest–if continuous, then this should be real values; if binary, then a vector of 0's and 1's; if count data, then a vector of integers.
X	An N x q matrix of the nonlinear predictors, with q either 1 or 2. All entries of X must be numeric real.
Z	An N x p matrix of linear predictors. All entries of X should be numeric; therefore, any categorical predictors should be converted into dummy variables before calling hpr.
X_aug	An N_aug x q matrix of nonlinear predictors at which interpolations/extrapolations are requested.
family	The family of the outcome; can be either "gaussian", "binomial", or "poisson". The default is "gaussian".
beta_dist	If a matrix Z is provided, this is the prior imposed on the coefficients of the linear predictors included in Z. Can be either "normal" or "cauchy". The default is "normal".
beta_mean	If a matrix Z is provided, then this is a vector of prior location parameters for the coefficients of the linear predictors. The default behavior will set all prior locations to be 0.
beta_sd	If a matrix Z is provided, then this is a vector of prior scale parameters for the coefficients of the linear predictors. The default behavior will set all prior scales to be 5 times the observed standard deviation for that predictor.

beta_df — If a matrix Z is provided and the beta distribution is "cauchy", this is a vector containing the degrees of freedom for each Cauchy prior for each of the coefficients of the linear predictors. The default behavior is to set beta_df to be 1 for all coefficients, which corresponds to a true Cauchy prior. Larger values of beta_df will yield t distributions with that many degrees of freedom.

monotonic_terms — A vector of less than length q containing the indices of the nonlinear predictors that should be constrained to be monotonic. See the vignette for more examples.

monotonic_approach — An indicator of which approach for constraining monotonicity should be used. The default is "abs" (the absolute value) which is recommended; users can also specify "exp" (the exponential function) which may have computational challenges.

aug_approach — An indicator of which approach for data augmentation should be used. The highly recommended default is "MCMC" which draws from the posterior to generate new augmentation values and local shrinkage parameters. Other options are "Mean" or "LVCF", which use the mean value of the two neighboring local shrinkage parameters, or the last value of the local shrinkage parameter to serve as the local shrinkage at the augmentation point, then inputs this value via Gaussian kriging equations. For more details, see Chase et al. (2022+). Again, we highly discourage the use of any approach other than "MCMC".

new_X — An N_new x q matrix of locations at which posterior predictions are requested. Note that all entries in this matrix must already appear in either X or X_aug in order for posterior predictions to be generated.

new_Z — An N_new x p matrix of linear predictor values at which posterior predictions are requested, corresponding to the nonlinear predictor values input in new_X.

seed — A number that will be passed to the Stan MCMC to make it possible to generate identical results on future runs.

chains — The number of chains used for Hamiltonian MCMC; the default is 4. We do not recommend modifying this parameter unless you know what you're doing.

iter_warmup — The number of samples for warmup per chain. These samples will not be included for posterior estimation. The default is 1000 (so 1000 x 4 chains = 4000 warmup samples). We do not recommend modifying this parameter unless you know what you're doing.

iter_sampling — The number of posterior samples per chain. The default is 2000 (so 2000 x 4 chains = 8000 posterior samples). We do not recommend modifying this parameter unless you know what you're doing.

max_treedepth — The max_treedepth parameter passed to the NUTS algorithm within the Hamiltonian MCMC. The default value is 12. We do not recommend modifying this parameter unless you know what you're doing.

adapt_delta — The adapt_delta parameter passed to the Hamiltonian MCMC, which helps to set the step size of the MCMC. The default value is 0.95. We do not recommend modifying this parameter unless you know what you're doing.

c — The scale parameter of the half-Cauchy prior placed on the global shrinkage parameter. The default is 0.01, which we have found yields reasonably good performance. In general, we find that this parameter does not make much difference except in the case of very sparse data, in which case changes in c can make a big difference. We recommend trying several values of c to explore sensitivity of results.

verbose — A logical value indicating whether or not you would like to print updates on Stan sampling progress. The default is verbose = FALSE.

Value

The function returns a named list with the following contents:

stan_object the finished Stan modeling object

run_time the length of time needed for HMC sampling, in seconds

model_file a character string containing the name of the Stan model file that was used

seed either the value of seed input by user, or the value of a seed that was randomly generated within hpr

grid the unique, ordered values of each column of X and X_aug

treedepth the value of max_treedepth as input by the user

adapt_delta the value of adapt_delta as input by the user

Examples

```
X <- as.matrix(dat$Day, ncol = 1)
y <- dat$Temperature

mymodel <- hpr(y = y, X = X, family = "gaussian")

mymodel_monotonic <- hpr(y = y, X = X, family = "gaussian",
monotonic_terms = c(1), monotonic_approach = "abs")

Z <- as.matrix(dat$Aberration, ncol = 1)

mymodel2 <- hpr(y = y, X = X, Z = Z, new_X = X, new_Z = Z,
family = "gaussian", beta_dist = "cauchy")
```

---

post_predict

Function to extract posterior predictions from an HPR model. Note that these are posterior predictions; if you want estimates of the estimated trajectory f (the systematic model component), use get_preds.

Description

Function to extract posterior predictions from an HPR model. Note that these are posterior predictions; if you want estimates of the estimated trajectory f (the systematic model component), use get_preds.

Usage

```
post_predict(object = NULL, alpha = 0.05)
```

Arguments

object — The results object from a run of hpr, for which a new_X and new_Z matrix were input.

alpha — The uncertainty level for posterior prediction intervals; the default is 0.05 (which corresponds to 95% prediction intervals).

Value

A dataframe with N_new rows (corresponding to the values of new_X, new_Z) and columns:

Mean the mean of the posterior prediction samples

Median the median of the posterior prediction samples

Lower the alpha/2 percentile of the posterior prediction samples

Upper the 1-alpha/2 percentile of the posterior prediction samples

Examples

```
X <- as.matrix(dat$Day, ncol = 1)
y <- dat$Temperature

mymodel <- hpr(y = y, X = X, family = "gaussian", new_X = X)
post_preds <- post_predict(mymodel)
post_preds$x <- dat$Day
```

Index