Package ‘shrinkTVP’

October 6, 2019

Type Package

Title Efficient Bayesian Inference for Time-Varying Parameter Models with Shrinkage

Version 1.1.1

Description
Efficient Markov chain Monte Carlo (MCMC) algorithms for fully Bayesian estimation of time-varying parameter models with shrinkage priors. Details on the algorithms used are provided in Bitto and Frühwirth-Schnatter (2019) <doi:10.1016/j.jeconom.2018.11.006>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0)

Imports Rcpp, GIGrvg, stochvol, coda, methods, utils, zoo

LinkingTo Rcpp, RcppArmadillo, GIGrvg, RcppProgress, stochvol

RoxygenNote 6.1.1

Suggests testthat, knitr, rmarkdown, R.rsp

VignetteBuilder R.rsp

NeedsCompilation yes

Author Peter Knaus [aut, cre] (<https://orcid.org/0000-0001-6498-7084>),
Angela Bitto-Nemling [aut],
Annalisa Cadonna [aut] (<https://orcid.org/0000-0003-0360-7628>),
Sylvia Frühwirth-Schnatter [aut]
(<https://orcid.org/0000-0003-0516-5552>),
Daniel Winkler [ctb],
Kemal Dingic [ctb]

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

Repository CRAN

Date/Publication 2019-10-06 16:00:02 UTC

R topics documented:

`eval_pred_dens` *Evaluate the one step ahead predictive density of a fitted TVP model*

Description

`eval_pred_dens` evaluates the one-step ahead predictive density of a fitted TVP model resulting from a call to shrinkTVP at the points supplied in `x`. For details on the approximation of the one-step ahead predictive density used, see the vignette.

Usage

```
eval_pred_dens(x, mod, data_test, log = FALSE)
```

Arguments

`x`	a real number or a vector of real numbers, taken to be the points at which the predictive density will be evaluated.
`mod`	an object of class `shrinkTVP`, containing the fitted model for which the predictive density should be evaluated.
`data_test`	a data frame with one row, containing the one step ahead covariates. The names of the covariates have to match the names of the covariates used during model estimation in the call to `shrinkTVP`.
`log`	a single logical value detrmining whether the density should be evaluated on the log scale or not.

Value

The value returned is a vector of length `length(x)`, containing the values of the predictive density evaluated at the points supplied in `x`.

Author(s)

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

Examples

```
# Simulate data
set.seed(123)
sim <- simTVP(theta = c(0.2, 0, 0), beta_mean = c(1.5, -0.3, 0))
data <- sim$data

# Estimate model
res <- shrinkTVP(y ~ x1 + x2, data = data[1:199, ])

# Create sequence of x values where the density is to be evaluated
x_vals <- seq(0, 12, by = 0.1)

# Evaluate density and plot
dens <- eval_pred_dens(x_vals, res, data[200, ])
plot(x_vals, dens, type = "l")

# Add vertical line where true value of the one step ahead y lies
abline(v = data$y[200])
```

`LPDS` *Calculate the Log Predictive Density Score for a fitted TVP model*

Description

`LPDS` calculates the one step ahead Log Predictive Density Score (LPDS) of a fitted TVP model resulting from a call to shrinkTVP. For details on the approximation of the one-step ahead predictive density used, see the vignette.

Usage

```
LPDS(mod, data_test)
```

Arguments

`mod`	an object of class `shrinkTVP`, containing the fitted model for which the LPDS should be calculated.
`data_test`	a data frame with one row, containing the one step ahead covariates and response. The names of the covariates and the response have to match the names used during model estimation in the call to `shrinkTVP`.

Value

A real number equaling the calculated LPDS.

Author(s)

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

Examples

```
# Simulate data
set.seed(123)
sim <- simTVP(theta = c(0.2, 0, 0), beta_mean = c(1.5, -0.3, 0))
data <- sim$data

# Estimate model
res <- shrinkTVP(y ~ x1 + x2, data = data[1:199, ])

# Calculate LPDS
LPDS(res, data[200,])
```

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

Description

`plot.mcmc.tvp` plots empirical posterior quantiles for a time-varying parameter.

Usage

```
## S3 method for class 'mcmc.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.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 defualt 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","dc` 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 `"red"`.
`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 `1`.
`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"` 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`>

Examples

```
set.seed(123)
sim <- simTVP(theta = c(0.2, 0, 0), beta_mean = c(1.5, -0.3, 0))
data <- sim$data

res <- shrinkTVP(y ~ x1 + x2, data)
plot(res$beta$beta_x1)
```

`plot.shrinkTVP` *Graphical summary of posterior distribution*

Description

`plot.shrinkTVP` generates plots visualizing the posterior distribution.

Usage

```
## S3 method for class 'shrinkTVP'
plot(x, pars = c("beta"), nplot = 3, mar = c(2,
  4, 1, 2) + 0.1, ...)
```

Arguments

`x`	a `shrinkTVP` 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 `shrinkTVP` 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.
`mar`	A numerical vector of the form `c(bottom,left,top,right)` which gives the number of lines of margin to be specified on the four sides of the plot, as in `par`. The default is c(2, 4, 1, 2) + 0.1.
`...`	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>

Examples

```
set.seed(123)
sim <- simTVP(theta = c(0.2, 0, 0), beta_mean = c(1.5, -0.3, 0))
data <- sim$data

output <- shrinkTVP(y ~ x1 + x2, data)
plot(output)


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

## End(Not run)
```

---

print.shrinkTVP *Nicer printing of shrinkTVP objects*

Description

Nicer printing of shrinkTVP objects

Usage

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

Arguments

x	A shrinkTVP object
...	Currently ignored.

Value

Called for its side effects and returns invisibly.

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

Description

shrinkTVP samples from the joint posterior distribution of the parameters of a time-varying parameter model with shrinkage, potentially including stochastic volatility (SV), and returns the MCMC draws.

Usage

```
shrinkTVP(formula, data, 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_kappa2 = TRUE, learn_lambda2 = TRUE,
  kappa2 = 20, lambda2 = 20, hyperprior_param, c_tuning_par_xi = 1,
  c_tuning_par_tau = 1, display_progress = TRUE, ret_beta_nc = FALSE,
  sv = FALSE, sv_param)
```

Arguments

formula	object of class "formula": a symbolic representation of the model, as in the function lm. For details, see formula.
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 shrinkTVP is called. No NAs are allowed in the response variable and the covariates.
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 local adaptation parameter of the state variances. The default value is TRUE.
learn_a_tau	logical value indicating whether to learn a_tau, the local adaptation 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. 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. The default value is 0.1.

`learn_kappa2` logical value indicating whether to learn kappa2, the global level of shrinkage for the state variances. The default value is `TRUE`.

`learn_lambda2` logical value indicating whether to learn the lambda^2 parameter, the global level of shrinkage for the mean of the initial values of the states. The default value is `TRUE`.

`kappa2` positive, real number, indicating the (fixed) value for kappa2. Ignored if `learn_kappa2` is `TRUE`. The default value is 20.

`lambda2` positive, real number, indicating the (fixed) value for lambda2. Ignored if `learn_lambda2` is `TRUE`. The default value is 20.

`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:

- `c0`: The default value is 2.5.
- `g0`: The default value is 5.
- `G0`: The default value is 5 / (2.5 - 1).
- `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.
- `b_xi`: The default value is 10.
- `b_tau`: The default value is 10.
- `nu_xi`: The default value is 5.
- `nu_tau`: The default value is 5.

`c_tuning_par_xi`

positive, real number. Determines the standard deviation of the proposal distribution for the Metropolis Hastings step for a_xi. Ignored if `learn_a_xi` is `FALSE`. The default value is 1.

`c_tuning_par_tau`

positive, real number. Determines the standard deviation of the proposal distribution for the Metropolis Hastings step for a_tau. Ignored if `learn_a_tau` is `FALSE`. The default value is 1.

`display_progress`

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

`ret_beta_nc` logical value indicating whether to output the non-centered states in addition to the centered ones. The default value is `FALSE`.

`sv` logical value indicating whether to use stochastic volatility for the error of the observation equation. For details please see `stochvol`, in particular `svsample`. The default value is `FALSE`.

`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 sv is FALSE. 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.
- bmu: real number. The default value is 0.
- Bmu: real number. larger than 0. The default value is 1.

Details

For details concerning the algorithm please refer to the paper by Bitto and Frühwirth-Schnatter (2019).

Value

The value returned is a list object of class shrinkTVP containing

sigma2	mcmc object containing the parameter draws from the posterior distribution of sigma2. If sv is TRUE, sigma2 is additionally an mcmc.tvp object.
theta_sr	an mcmc object containing the parameter draws from the posterior distribution of the square root of theta.
beta_mean	an mcmc object containing the parameter draws from the posterior distribution of beta_mean.
beta_nc	*(optional)* list object containing an mcmc.tvp object for the parameter draws from the posterior distribution of the non-centered states, one for each covariate. In the case that there is only one covariate, this becomes just a single mcmc.tvp object. Not returned if ret_beta_nc is FALSE.
beta	list object containing an mcmc.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.tvp object.
xi2	mcmc object containing the parameter draws from the posterior distribution of xi2.
a_xi	*(optional)* mcmc object containing the parameter draws from the posterior distribution of a_xi. Not returned if learn_a_xi is FALSE.
a_xi_acceptance	*(optional)* list object containing acceptance statistics for the Metropolis Hastings algorithm for a_xi. Not returned if learn_a_xi is FALSE.
tau2	mcmc object containing the parameter draws from the posterior distribution of tau2.
a_tau	*(optional)* mcmc object containing the parameter draws from the posterior distribution of a_tau. Not returned if learn_a_tau is FALSE.
a_tau_acceptance	*(optional)* list containing acceptance statistics for the Metropolis Hastings algorithm for a_tau. Not returned if learn_a_tau is FALSE.
kappa2	*(optional)* mcmc object containing the parameter draws from the posterior distribution of kappa2. Not returned if learn_kappa2 is FALSE.

`lambda2`	*(optional)* `mcmc` object containing the parameter draws from the posterior distribution of lambda2. Not returned if `learn_lambda2` is `FALSE`.
`C0`	*(optional)* `mcmc` object containing the parameter draws from the posterior distribution of C0. Not returned if `sv` is `TRUE`.
`sv_mu`	*(optional)* `mcmc` object containing the parameter draws from the posterior distribution of the mu parameter for the stochastic volatility model on the errors. Not returned if `sv` is `FALSE`.
`sv_phi`	*(optional)* `mcmc` object containing the parameter draws from the posterior distribution of the phi parameter for the stochastic volatility model on the errors. Not returned if `sv` is `FALSE`.
`sv_sigma2`	*(optional)* `mcmc` object containing the parameter draws from the posterior distribution of the sigma2 parameter for the stochastic volatility model on the errors. Not returned if `sv` is `FALSE`.
`priorvals`	`list` object containing hyperparameter values of the prior distributions, as specified by the user.
`model`	`list` object containing the model matrix and model response used.
`summaries`	`list` object containing a collection of summary statistics of the posterior draws.
`LPDS_comp`	`list` object containg two arrays that are required for calculating the LPDS.

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.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)

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

References

Bitto, A., & Frühwirth-Schnatter, S. (2019). "Achieving shrinkage in a time-varying parameter model framework." *Journal of Econometrics*, 210(1), 75-97. <doi:10.1016/j.jeconom.2018.11.006>

See Also

`plot.shrinkTVP`, `plot.mcmc.tvp`

Examples

```
## Example 1, learn everything
set.seed(123)
sim <- simTVP(theta = c(0.2, 0, 0), beta_mean = c(1.5, -0.3, 0))
data <- sim$data

res <- shrinkTVP(y ~ x1 + x2, data = data)
```

```
# summarize output
summary(res)


## Example 2, hierarchical Bayesian Lasso
res <- shrinkTVP(y ~ x1 + x2, data = data,
                learn_a_xi = FALSE, learn_a_tau = FALSE,
                a_xi = 1, a_tau = 1)


## Example 3, non-hierarchical Bayesian Lasso
res <- shrinkTVP(y ~ x1 + x2, data = data,
                learn_a_xi = FALSE, learn_a_tau = FALSE,
                a_xi = 1, a_tau = 1,
                learn_kappa2 = FALSE, learn_lambda2 = FALSE)


## Example 4, adding stochastic volatility
res <- shrinkTVP(y ~ x1 + x2, data = data,
                sv = TRUE)


## Example 4, changing some of the default hyperparameters
res <- shrinkTVP(y ~ x1 + x2, data = data,
                hyperprior_param = list(b_xi = 5,
                                        nu_xi = 10))
```

simTVP *Generate synthetic data from a time-varying parameter model*

Description

simTVP generates synthetic data from a time-varying parameter model. The covariates are always generated i.i.d. from a Normal(0,1) distribution.

Usage

```
simTVP(N = 200, d = 3, sv = FALSE, sigma2 = 1, theta, beta_mean)
```

Arguments

N	integer > 2. Indicates the length of the time series to be generated. The default value is 200.
d	positive integer. Indicates the number of covariates to simulate. The default value is 3.
sv	logical value. If set to TRUE, the data will be generated with stochastic volatility for the errors of the observation equation using svsim. The default value is FALSE.

`sigma2`	positive real number. Determines the variance on the errors of the observation equation. Ignored if sv is `TRUE`. The default value is 1.
`theta`	*(optional)* vector containing positive real numbers. If supplied, these determine the variances of the innovations of the state equation. Otherwise, the elements of `theta` are generated from a ChiSq(1) distribution. Has to be of length `d` or an error will be thrown.
`beta_mean`	*(optional)* vector containing real numbers. If supplied, these determine the mean of the initial value of the state equation. Otherwise, the elements of `beta_mean` are generated from a Normal(0,1) distribution. Has to be of length `d` or an error will be thrown.

Value

The value returned is a list object containing:

`data`	a data frame that holds the simulated data.
`true_vals`	a list object containing: • `theta`: the values of theta used in the data generating process. • `beta_mean`: the values of beta_mean used in the data generating process. • `beta`: the true paths of beta used for the data generating process. • `sigma2`: the value(s) of sigma2 used in the data generating process.

Author(s)

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

Examples

```
# Generate a time series of length 300
res <- simTVP(N = 300)

# Extract the generated data
data <- res$data

# Now with stochastic volatility
res_sv <- simTVP(N = 300, sv = TRUE)
```

Index