Package 'shrinkTVP'

November 9, 2020

Type Package

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

Version 2.0.1

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> and Cadonna et al. (2020) <doi:10.3390/econometrics8020020>.

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 7.1.1

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 2020-11-09 21:40:06 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>

See Also

Other prediction functions: LPDS(), fitted.shrinkTVP(), forecast_shrinkTVP(), predict.shrinkTVP(), residuals.shrinkTVP()

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])
```

---

fitted.shrinkTVP *Calculate fitted historical values for an estimated TVP model*

Description

Calculates the fitted values for an estimated TVP model, i.e. $X_t'\beta_t$. Note that in contrast to predict.shrinkTVP this does not include the error term.

Usage

```
## S3 method for class 'shrinkTVP'
fitted(object, ...)
```

Arguments

object	A shrinkTVP object
...	Currently ignored.

Value

An object of class shrinkTVP_fitted

Author(s)

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

See Also

Other prediction functions: LPDS(), eval_pred_dens(), forecast_shrinkTVP(), predict.shrinkTVP(), residuals.shrinkTVP()

Examples

```
# Generate synthetic data
sim <- simTVP()

# Estimate a model
res <- shrinkTVP(y ~ x1 + x2, sim$data)

# Calculate fitted values
fitted <- fitted(res)

# Visualize
plot(fitted)
lines(sim$data$y, col = "forestgreen")
```

---

forecast_shrinkTVP *Draw from posterior predictive density of a fitted TVP model*

Description

forecast_shrinkTVP draws from the posterior predictive distribution of a fitted TVP model resulting from a call to shrinkTVP.

Usage

```
forecast_shrinkTVP(mod, newdata, n.ahead)
```

Arguments

mod	an object of class shrinkTVP, containing the fitted model.
newdata	a data frame containing the future covariates. The names of the covariates have to match the names used during model estimation in the call to shrinkTVP.
n.ahead	a single, positive integer indicating the forecasting horizon, i.e. how many time-points into the future the posterior predictive distribution should be sampled from. Can not be larger than the number of rows in newdata.

Value

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

Author(s)

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

See Also

Other prediction functions: `LPDS()`, `eval_pred_dens()`, `fitted.shrinkTVP()`, `predict.shrinkTVP()`, `residuals.shrinkTVP()`

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[1:190, ])

# Forecast
forc <- forecast_shrinkTVP(res, data[191:200, ])

# Plot
plot(forc)
```

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>

See Also

Other prediction functions: eval_pred_dens(), fitted.shrinkTVP(), forecast_shrinkTVP(), predict.shrinkTVP(), residuals.shrinkTVP()

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"`, `"do` 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>`

See Also

Other plotting functions: `plot.shrinkTVP_forc()`, `plot.shrinkTVP()`

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,
  h_borders = c(0.05, 0.05),
  w_borders = c(0.02, 0.02),
  ...
)
```

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.
`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.tvp(), plot.shrinkTVP_forc()

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)
```

---

plot.shrinkTVP_forc *Graphical summary of posterior predictive density*

Description

plot.shrinkTVP_forc generates plots visualizing the posterior predictive density generated by forecast_shrinkTVP.

Usage

```
## S3 method for class 'shrinkTVP_forc'
plot(x, showgap = FALSE, ...)
```

Arguments

x	a shrinkTVP_forc object.
showgap	if showgap = FALSE, the gap between the historical observations and the forecasts is removed. The default value is FALSE.
...	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.mcmc.tvp(), plot.shrinkTVP()

Examples

```
set.seed(123)
sim <- simTVP()

train <- sim$data[1:190, ]
test <- sim$data[191:200, ]

res <- shrinkTVP(y ~ x1 + x2, train)

forecast <- forecast_shrinkTVP(res, test)
plot(forecast)
lines(sim$data$y, col = "forestgreen")
```

---

predict.shrinkTVP *Calculate predicted historical values for an estimated TVP model*

Description

Calculates the predicted past values for an estimated TVP model, i.e. $X_t'\beta_t + \epsilon_t$. Note that in contrast to fitted.shrinkTVP this includes the error term.

Usage

```
## S3 method for class 'shrinkTVP'
predict(object, ...)
```

Arguments

object	a shrinkTVP object
...	Currently ignored.

Value

An object of class shrinkTVP_pred.

Author(s)

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

See Also

Other prediction functions: LPDS(), eval_pred_dens(), fitted.shrinkTVP(), forecast_shrinkTVP(), residuals.shrinkTVP()

Examples

```
# Generate synthetic data
sim <- simTVP(N = 300)

# Estimate a model
res <- shrinkTVP(y ~ x1 + x2, sim$data)

# Calculate predicted values
pred <- predict(res)

# Visualize
plot(pred)
lines(sim$data$y, col = "forestgreen")
```

---

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.

Author(s)

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

`residuals.shrinkTVP` *Calculate residuals for an estimated TVP model*

Description

Calculates the residuals for an estimated TVP model, i.e. $y_t - X_t'\beta_t$.

Usage

```
## S3 method for class 'shrinkTVP'
residuals(object, ...)
```

Arguments

`object`	a `shrinkTVP` object.
`...`	Currently ignored.

Value

An object of class `shrinkTVP_resid`

Author(s)

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

See Also

Other prediction functions: `LPDS()`, `eval_pred_dens()`, `fitted.shrinkTVP()`, `forecast_shrinkTVP()`, `predict.shrinkTVP()`

Examples

```
# Generate synthetic data
sim <- simTVP(N = 300)

# Estimate a model
res <- shrinkTVP(y ~ x1 + x2, sim$data)

# Calculate residuals
resids <- residuals(res)

# Visualize
plot(resids)
```

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,
  mod_type = "double",
  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,
  display_progress = TRUE,
  sv = FALSE,
  sv_param,
  MH_tuning,
  starting_vals
)
```

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.

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".

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. Ignored if mod_type is set to "ridge". 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. Ignored if mod_type is set to "ridge". 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, indicating the (fixed) value for kappa2_B. Ignored if `learn_kappa2_B` is `TRUE`. The default value is 20.

`lambda2_B`
: positive, real number, indicating the (fixed) value for lambda2_B. Ignored if `learn_lambda2_B` 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.
  - `alpha_a_xi`: The default value is 5.
  - `alpha_a_tau`: The default value is 5.
  - `beta_a_xi`: The default value is 10.
  - `beta_a_tau`: The default value is 10.
  - `alpha_c_xi`: The defalut value is 5.
  - `alpha_c_tau`: The defalut value is 5.
  - `beta_c_xi`: The default value is 2.
  - `beta_c_tau`: The default value is 2.

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

`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.

MH_tuning *optional* named list containing values used to tune the MH steps for a_xi, a_tau, c_xi and c_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 and mod_type is not equal to "ridge". The arguments for c_xi(c_tau) are only used if learn_c_xi(learn_c_tau) is set to TRUE and mod_type is equal to "triple". 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.
- c_xi_adaptive: logical value. The default is TRUE.
- c_xi_tuning_par: positive, real number. The default value is 1.
- c_xi_target_rate: positive, real number, between 0 and 1. The default value is 0.44.
- c_xi_max_adapt: positive, real number. The default value is 0.01.
- c_xi_batch_size: positive integer. The default value is 50.
- c_tau_adaptive: logical value. The default is TRUE.
- c_tau_tuning_par: positive, real number. The default value is 1.
- c_tau_target_rate: positive, real number, between 0 and 1. The default value is 0.44.
- c_tau_max_adapt: positive, real number. The default value is 0.01.
- c_tau_batch_size: positive integer. The default value is 50.

starting_vals *optional* named list containing the values at which the MCMC algorithm will be initialized. In the following d refers to the number of covariates, including the intercept and expanded factors. 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 following elements can be supplied:

- `beta_mean_st`: vector of length `d` containing single numbers. The default is `rep(0,d)`.
- `theta_sr_st`: vector of length `d` containing single, positive numbers. The default is `rep(1,d)`.
- `tau2_st`: vector of length `d` containing single, positive numbers. The default is `rep(1,d)`.
- `xi2_st`: vector of length `d` containing single, positive numbers. The default is `rep(1,d)`.
- `kappa2_st`: vector of length `d` containing single, positive numbers. The default is `rep(1,d)`.
- `lambda2_st`: vector of length `d` containing single, positive numbers. The default is `rep(1,d)`.
- `kappa2_B_st`: positive, real number. The default value is 20.
- `lambda2_B_st`: positive, real number. The default value is 20.
- `a_xi_st`: positive, real number. The default value is 0.1.
- `a_tau_st`: positive, real number. The default value is 0.1.
- `c_xi_st`: positive, real number. The default value is 0.1. Note that the prior for `c_xi` is restricted to (0, 0.5).
- `c_tau_st`: positive, real number. The default value is 0.1. Note that the prior for `c_tau` is restricted to (0, 0.5).
- `sv_mu_st`: real number. The default value is -10.
- `sv_phi_st`: positive, real number between -1 and 1. The default value is 0.5.
- `sv_sigma2_st` : positive, real number. The default value is 1.
- `C0_st`: positive, real number. The default value is 1.
- `sigma2_st`: positive, real number if `sv` is `FALSE`, otherwise a vector of positive, real numbers of length `N`. The default value is 1 or a vector thereof.
- `h0_st`: real number. The default value is 0.

Details

For details concerning the algorithms please refer to the papers by Bitto and Frühwirth-Schnatter (2019) and Cadonna et al. (2020).

Value

The value returned is a list object of class `shrinkTVP` containing

`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.
`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".

sigma2 — mcmc object containing the parameter draws from the posterior distribution of sigma2. If sv is TRUE, sigma2 is additionally an mcmc.tvp object.

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.

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.

internals — list object containg two arrays that are required for calculating the LPDS.

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.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>

Cadonna, A., Frühwirth-Schnatter, S., & Knaus, P. (2020). "Triple the Gamma—A Unifying Shrinkage Prior for Variance and Variable Selection in Sparse State Space and TVP Models." *Econometrics*, 8(2), 20. <doi:10.3390/econometrics8020020>

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(beta_a_xi = 5,
                                         alpha_a_xi = 10))

## Example 5, using the triple gamma prior
res <- shrinkTVP(y ~ x1 + x2, data = data,
                 mod_type = "triple")
```

---

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,
  sv_mu = 0,
  sv_phi = 0.98,
  sv_sigma2 = 0.2
)
```

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 of 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 X^2(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.
sv_mu	real number. Determines the mean of the logarithm of the volatility. Ignored if sv is FALSE. The default value is 0.
sv_phi	real number between -1 and 1. Determines the persistence of the SV process. Ignored if sv is FALSE. The default value is 0.98.
sv_sigma2	positive, real number. Determines the variance of the innovations of the logarithm of the volatility. Ignored if sv is FALSE. The default value is 0.2.

Value

The value returned is a list object containing:

data	data frame that holds the simulated data.
true_vals	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)
```

updateTVP *One step update version of* shrinkTVP *with minimal overhead*

Description

updateTVP draws a single sample from the joint posterior distribution of the parameters of a time-varying parameter model with shrinkage potentially including stochastic volatility (SV). It performs **no input checks** and must therefore be used with caution. It is designed to be used in a modular fashion within other samplers, where speed is important. As such, no draws are saved and must be stored manually if the user wants to analyze them further.

Usage

```
updateTVP(
  y,
  x,
  curr_draws,
  mod_type = "double",
  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 = FALSE,
  sv_param,
  MH_tuning
)
```

Arguments

y	vector of length N containing the response variable.
x	matrix of dimension Nxd containing the covariates.
curr_draws	named list containing all the current draws from the joint posterior of the parameters. Not all values are required for all model setups. The following elements can be supplied: • beta_mean_st: vector of length d containing single numbers.

- theta_sr_st: vector of length d containing single, positive numbers.
- tau2_st: *optional* vector of length d containing single, positive numbers. Not required if mod_type is "ridge".
- xi2_st: *optional* vector of length d containing single, positive numbers. Not required if mod_type is "ridge".
- kappa2_st: *optional* vector of length d containing single, positive numbers. Only required if mod_type is "triple".
- lambda2_st: *optional* vector of length d containing single, positive numbers. Only required if mod_type is "triple".
- kappa2_B_st: *optional* positive, real number. Not required if mod_type is "ridge" or learn_kappa2_B is FALSE.
- lambda2_B_st: *optional* positive, real number. Not required if mod_type is "ridge" or learn_lambda2_B is FALSE.
- a_xi_st: *optional* positive, real number. Not required if mod_type is "ridge" or learn_a_xi is FALSE.
- a_tau_st: *optional* positive, real number. Not required if mod_type is "ridge" or learn_a_tau is FALSE.
- c_xi_st: *optional* positive, real number. Note that the prior for c_xi is restricted to (0, 0.5). Not required if mod_type is not "triple" or learn_c_xi is FALSE.
- c_tau_st: *optional* positive, real number. Note that the prior for c_tau is restricted to (0, 0.5). Not required if mod_type is not "triple" or learn_c_tau is FALSE.
- sv_mu_st: *optional* real number. Not required if sv is FALSE.
- sv_phi_st: *optional* positive, real number between -1 and 1. Not required if sv is FALSE.
- sv_sigma2_st : *optional* positive, real number. Not required if sv is FALSE.
- C0_st: *optional* positive, real number. Not required if sv is TRUE.
- sigma2_st: positive, real number if sv is FALSE, otherwise a vector of positive, real numbers of length N. The default value is 1 or a vector thereof.
- h0_st: *optional* real number. The default value is 0. Not required if sv is FALSE.

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".

learn_a_xi — logical value indicating whether to learn a_xi, the spike parameter of the state variances. Ignored if mod_type is set to "ridge". 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. Ignored if mod_type is set to "ridge". 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". 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". The default value is TRUE.

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

a_eq_c_xi — logical value indicating whether to force a_xi and c_xi to be equal. 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. 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, indicating the (fixed) value for kappa2_B. Ignored if learn_kappa2_B is TRUE. The default value is 20.

lambda2_B — positive, real number, indicating the (fixed) value for lambda2_B. Ignored if learn_lambda2_B 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.
- 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.
- beta_c_xi: The default value is 2.
- alpha_c_xi: The defalut value is 5.
- beta_c_tau: The default value is 2.

- alpha_c_tau: The defalut value is 5.

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.

MH_tuning — *optional* named list containing values used to tune the MH steps for a_xi, a_tau, c_xi and c_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 and mod_type is not equal to "ridge". The arguments for c_xi(c_tau) are only used if learn_c_xi(learn_c_tau) is set to TRUE and mod_type is equal to "triple". 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.
- c_xi_adaptive: logical value. The default is TRUE.
- c_xi_tuning_par: positive, real number. The default value is 1.

- c_xi_target_rate: positive, real number, between 0 and 1. The default value is 0.44.
- c_xi_max_adapt: positive, real number. The default value is 0.01.
- c_xi_batch_size: positive integer. The default value is 50.
- c_tau_adaptive: logical value. The default is TRUE.
- c_tau_tuning_par: positive, real number. The default value is 1.
- c_tau_target_rate: positive, real number, between 0 and 1. The default value is 0.44.
- c_tau_max_adapt: positive, real number. The default value is 0.01.
- c_tau_batch_size: positive integer. The default value is 50.

Value

The value returned is a named list object which can be immediately used as the curr_draws argument for another draw from the posterior with updateTVP. Note that, depending on the model setup, some elements may be matrices of dimension zero. It contains the following elements:

beta_st	dx(N + 1) matrix containing the current draw from the posterior distribtuion of beta.
beta_mean_st	dx1 matrix containing the current draws from the posterior distribution of beta_mean.
theta_sr_st	dx1 matrix containing the current draws from the posterior distribution of the square root of theta.
tau2_st	dx1 matrix containing the current draws from the posterior distribution of tau2.
xi2_st	dx1 matrix containing the current draws from the posterior distribution of xi2.
lambda2_st	dx1 matrix containing the current draws from the posterior distribution of lambda2.
kappa2_st	dx1 matrix containing the current draws from the posterior distribution of kappa2.
a_xi_st	number representing the current draw from the posterior distribution of a_xi.
a_tau_st	number representing the current draw from the posterior distribution of a_tau.
c_xi_st	number representing the current draw from the posterior distribution of c_xi.
c_tau_st	number representing the current draw from the posterior distribution of c_tau.
lambda2_B_st	number representing the current draw from the posterior distribution of lambda2_B.
kappa2_B_st	mcmc object containing the parameter draws from the posterior distribution of kappa2_B.
sigma2_st	number if sv is FALSE, otherwise a vector of length N containing the current draws from the posterior distribution of sigma2.
C0_st	number representing the current draw from the posterior distribution of C0.
sv_mu_st	number representing the current draw from the posterior distribution of the mu parameter for the stochastic volatility model on the errors.
sv_phi_st	number representing the current draw from the posterior distribution of the phi parameter for the stochastic volatility model on the errors.
sv_sigma2_st	number representing the current draw from the posterior distribution of the sigma2 parameter for the stochastic volatility model on the errors.

h0_st	number representing the current draw from the posterior distribution of the h0 parmater for the stochastic volatility model on the errors.
internals	list object containg two arrays that are required for calculating the LPDS and bookkeeping objects required for the adaptive MH algorithm to work.

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>

Cadonna, A., Frühwirth-Schnatter, S., & Knaus, P. (2020). "Triple the Gamma—A Unifying Shrinkage Prior for Variance and Variable Selection in Sparse State Space and TVP Models." *Econometrics*, 8(2), 20. <doi:10.3390/econometrics8020020>

Examples

```
# Simulate data
sim <- simTVP()
y <- sim$data$y
x <- as.matrix(sim$data[,2:4])

# Create starting values
d <- ncol(x)
curr_draws <- list(beta_mean_st = rep(0, d),
                   theta_sr_st = rep(1, d),
                   tau2_st = rep(1, d),
                   xi2_st = rep(1, d),
                   lambda2_st = rep(1, d),
                   kappa2_B_st = 20,
                   lambda2_B_st = 20,
                   a_xi_st = 0.1,
                   a_tau_st = 0.1,
                   c_tau_st = 0.1,
                   sv_mu_st = -10,
                   sv_phi_st = 0.5,
                   sv_sigma2_st = 1,
                   C0_st = 1,
                   sigma2_st = 1,
                   h0_st = 0)

# Run the algorithm for 1000 iterations
# Note that curr_draws is always re-written and immediately re-used
for (i in 1:1000){
  curr_draws <- updateTVP(y, x, curr_draws)
}
```

Index