Package ‘tvReg’

November 27, 2017

Type Package
Title Time-Varying Coefficients Linear Regression for Single and Multiple Equations
Version 0.2.1
Date 2017-11-02
Author Isabel Casas and Ruben Fernandez-Casal
Maintainer Isabel Casas <casasis@gmail.com>
Description Fitting simultaneous equations with time varying coefficients, both for the case of independent equations and for the case of correlated equations.
License GPL (>= 3)
LazyData yes
Depends R (>= 3.0.1), Matrix, graphics, stats (>= 2.14.0)
Imports systemfit (>= 1.1-20), MASS, vars, bvarsv
Suggests knitr, rmarkdown
RoxygenNote 6.0.1
VignetteBuilder knitr
NeedsCompilation no
Repository CRAN
Date/Publication 2017-11-27 11:54:07 UTC

R topics documented:

  bw ................................................................. 2
  bwCov ......................................................... 3
  CEES .......................................................... 4
  CI ............................................................... 5
  FF5F ............................................................ 6
  plot.tvsure ................................................... 8
  tvAcoef ....................................................... 9
  tvAR .......................................................... 10
Description

Calculate bandwidth(s) by cross-validation for functions tvSURE, tvVAR and tvLM.

Usage

bw(x, ...)  

## Default S3 method:  
bw(x, y, z = NULL, est = c("lc", "ll"),  
    tkernel = c("Epa", "Gaussian"), singular.ok = TRUE, ...)

## S3 method for class 'list'  
bw(x, y, z = NULL, est = c("lc", "ll"), tkernel = c("Epa",  
    "Gaussian"), singular.ok = TRUE, ...)

Arguments

x an object used to select a method.

... Other parameters passed to specific methods.

y A matrix or vector with the dependent variable(s).

z A vector with the variable over which coefficients are smooth over.

est The nonparametric estimation method, one of "lc" (default) for linear constant  
or "ll" for local linear.

tkernel The type of kernel used in the coefficients estimation method, one of Epanes-

nikov ("Epa") or "Gaussian".

singular.ok Logical. If FALSE, a singular model is an error.
Value

bw returns a vector or a scalar with the bandwidth to estimate the mean or the covariance residuals, fitted values.

A scalar or a vector of scalars.

Examples

tau <- seq(1:200)/200eta <- data.frame(beta1 = sin(2*pi*tau), beta2 = 2*tau)
X <- data.frame(X1 = rnorm(200), X2 = rchisq(200, df = 4))
error <- rt(200, df = 10)
y <- apply(X*beta, 1, sum) + error
bw <- bw(X, y, est = "ll", tkernel = "Gaussian")
model.tv <- tvOLS(x = X, y = y, bw = bw)
data(Kmenta, package = "systemfit")## x is a list of matrices containing the regressors, one matrix for each equation
x <- list()
x[[1]] <- Kmenta[, c("price", "income")]
x[[2]] <- Kmenta[, c("price", "FarmPrice", "trend")]
## y is a matrix with one column for each equation
y <- cbind(Kmenta$consump, Kmenta$consump)
## Calculate bandwidth
bw <- bw(x = x, y = y)
## One bandwidth per equation
print(bw)
## Use these bandwidths to estimate the time-varying coefficients
tvgs <- tvGLS(x = x, y = y, bw = bw)

---

bwCov

Covariance Bandwidth Calculation by Cross-Validation bwCov calculates a single bandwidth to estimate the time-varying variance-covariance matrix.

Description

Covariance Bandwidth Calculation by Cross-Validation bwCov calculates a single bandwidth to estimate the time-varying variance-covariance matrix.

Usage

bwCov(x, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"))
Arguments

- `x`: A matrix or a data frame.
- `est`: The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
- `tkernel`: The type of kernel used in the coefficients estimation method, one of Epanesnikov ("Epa") or "Gaussian".

Value

A scalar.

Examples

```r
data(CEES)
## Using a shorter set for a quick example
mydata <- tail(CEES, 50)
bw.cov <- bwCov(mydata)
Sigma.hat <- tvCov(mydata, bw = bw.cov)
```

Description

Aslanidis and Casas (2013) consider a portfolio of daily US dollar exchange rates of the Australian dollar (AUS), Swiss franc (CHF), euro (EUR), British pound (GBP), South African rand (RAND), Brazilian real (REALB), and Japanese yen (YEN) over the period from January 1, 1999 until May 7, 2010 (T=2856 observations). This dataset contains the standardised rates after "devolatilisation", i.e. standardising the rates using a GARCH(1,1) estimate of the volatility.

Format

A data frame with 2856 rows and 7 variables. Below the standardised rates of daily US dollar exchange rates of

- **AUS**: Australian dollar
- **CHF**: Swiss franc
- **EUR**: Euro
References


---

**Description**

CI is used to estimate the bootstrap confidence intervals for objects with class attribute `tvlm`, `tvvar`, `tvirf`, `tvsure`.

**Usage**

```r
CI(object, ...)  
## Default S3 method:
CI(object, level = 0, runs = 0, tboot = NULL, ...)

## S3 method for class 'tvirf'
CI(object, level = 0, runs = 0, tboot = NULL, ...)
```

**Arguments**

- `object` Object of class `tvsure`, class `tvvar` or class `tvirf`.
- `...` Other parameters passed to specific methods.
- `level` Numeric, the confidence level required (between 0 and 1).
- `runs` (optional) Number of bootstrap replications.
- `tboot` Type of wild bootstrap, choices `’wild’`(default), `’wild2’`. Option `’wild’` uses the distribution suggested by Mammen (1993) in the wild resampling, while `’wild2’` uses the standard normal.

**Value**

an object of class `tvsure` with BOOT, Lower and Upper different from NULL.
References


See Also

tvlM, tvAR, tvVAR, tvSURE

Examples

```r
# Calculation of confidence intervals for a tvLM model
tau <- seq(1:500)/500
beta <- data.frame(beta1 = sin(2*pi*tau), beta2 = 2*tau)
X1 <- rnorm(500)
X2 <- rchisq(500, df = 4)
error <- rt(500, df = 10)
y <- apply(cbind(X1, X2)*beta, 1, sum) + error
data <- data.frame(y = y, X1 = X1, X2 = X2)
model.tvlm <- tvLM(y~0+X1+X2, data = data, bw = 0.1)
tvci <- ci(model.tvlm, level = 0.95, runs = 30)
plot(tvci)
```

FF5F  
*Fama and French portfolio excess returns and factors for international markets.*

Description

A dataset containing the excess returns of four portfolios ordered by size and book-to-market. The four portfolios are SMALL/LoBM, SMALL/HiBM, BIG/LoBM and BIG/HiBM in four international market: North America (NA), Japan (JP), Asia Pacific (AS) and Europe (EU). It also contains the Fama/French 5 factors for each of the markets.

Format

A data frame with 314 rows and 41 variables.

Date Date, months from July 1990 until August 2016

NA.SMALL.LoBM  Excess return of portfolio SMALL/LoBM in North American market

NA.SMALL.HiBM  Excess return of portfolio SMALL/HiBM in North American market

NA.BIG.LoBM  Excess return of portfolio BIG/LoBM in North American market
NA.BIG.HiBM  Excess return of portfolio BIG/HiBM in North American market
NA.Mkt.RF  North American market excess return
NA.SMB  SMB (Small Minus Big) for the North American market
NA.HML  HML (High Minus Low) for the North American market
NA.RMW  RMW (Robust Minus Weak) for the North American market
NA.CMA  CMA (Conservative Minus Aggressive) for the North American market
NA.RF  North American risk free rate

JP.SMALL.LoBM  Excess return of portfolio SMALL/LoBM in Japanese market
JP.SMALL.HiBM  Excess return of portfolio SMALL/HiBM in Japanese market
JP.BIG.LoBM  Excess return of portfolio BIG/LoBM in Japanese market
JP.BIG.HiBM  Excess return of portfolio BIG/HiBM in Japanese market
JP.Mkt.RF  Japanese market excess return
JP.SMB  SMB (Small Minus Big) for the Japanese market
JP.HML  HML (High Minus Low) for the Japanese market
JP.RMW  RMW (Robust Minus Weak) for the Japanese market
JP.CMA  CMA (Conservative Minus Aggressive) for the Japanese market
JP.RF  Japanese risk free rate

AP.SMALL.LoBM  Excess return of portfolio SMALL/LoBM in Asia Pacific market
AP.SMALL.HiBM  Excess return of portfolio SMALL/HiBM in Asia Pacific market
AP.BIG.LoBM  Excess return of portfolio BIG/LoBM in Asia Pacific market
AP.BIG.HiBM  Excess return of portfolio BIG/HiBM in Asia Pacific market
AP.Mkt.RF  Asia Pacific market excess return
AP.SMB  SMB (Small Minus Big) for the Asia Pacific market
AP.HML  HML (High Minus Low) for the Asia Pacific market
AP.RMW  RMW (Robust Minus Weak) for the Asia Pacific market
AP.CMA  CMA (Conservative Minus Aggressive) for the Asia Pacific market
AP.RF  Asia Pacific risk free rate

EU.SMALL.LoBM  Excess return of portfolio SMALL/LoBM in European market
EU.SMALL.HiBM  Excess return of portfolio SMALL/HiBM in European market
EU.BIG.LoBM  Excess return of portfolio BIG/LoBM in European market
EU.BIG.HiBM  Excess return of portfolio BIG/HiBM in European market
EU.Mkt.RF  European market excess return
EU.SMB  SMB (Small Minus Big) for the European market
EU.HML  HML (High Minus Low) for the European market
EU.RMW  RMW (Robust Minus Weak) for the European market
EU.CMA  CMA (Conservative Minus Aggressive) for the European market
EU.RF  European risk free rate
Source

http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html

References


---

**plot.tvsure**

*Plot Methods for objects in tvReg*

**Description**

Plot methods for objects with class attribute tvlm, tvar, tvvar, tvirf, tvsure.

**Usage**

```r
## S3 method for class 'tvsure'
plot(x, eqs = NULL, vars = NULL, ...)

## S3 method for class 'tvlm'
plot(x, ...)

## S3 method for class 'tvar'
plot(x, ...)

## S3 method for class 'tvvar'
plot(x, ...)

## S3 method for class 'tvirf'
plot(x, obs.index = NULL, impulse = NULL, response = NULL,
     plot.type = c("multiple", "single"), names = NULL, main = NULL,
     sub = NULL, ylab = NULL, xlab = NULL, nc, mar.multi = c(0, 4, 0, 4),
     oma.multi = c(6, 4, 6, 4), adj.mtext = NA, padj.mtext = NA, ...)
```

**Arguments**

- `x` An x used to select a method.
- `eqs` Character vector (optional) with the equation(s) number(s) or equation name(s) of the coefficients to be plotted.
- `vars` Character vector (optional) with the variable number(s) or variable name(s) of the coefficients to be plotted.
- `...` Other parameters passed to specific methods.
obs.index  Scalar (optional), the time at which the impulse response is plotted. If left NULL, the mean over the whole period is plotted (this values should be similar to the estimation using a non time-varying VAR method).

impulse  Character vector (optional) of the impulses, default is all variables.

response  Character vector (optional) of the responses, default is all variables.

plot.type  Character, if multiple all plots are drawn in a single device, otherwise the plots are shown consecutively.

names  Character vector (optional), the variables names to be plotted. If left NULL, all variables are plotted.

main  Character vector, the titles of the plot.

sub  Character, sub title in plot.

ylab  Character vector signifying the labels for the y-axis.

xlab  Character vector signifying the labels for the x-axis.

nc  Integer, number of columns for multiple plot.

mar.multi  Setting of margins, if plot.type = "multiple".

oma.multi  Setting of margins, if plot.type = "multiple".

adj.mtext  Adjustment for mtext(), only applicable if plot.type = "multiple".

padj.mtext  Adjustment for mtext(), only applicable if plot.type = "multiple".

See Also

tvLM, tvVAR, tvSURE

tvAcoef  Time-Varying Coefficient Arrays of the Lagged Endogenous Variables of a tv-VAR (no intercept).

Description

Returns the estimated coefficients of the lagged endogenous variables as an array. Given an estimated time varying VAR of the form:

\[ \hat{y}_t = \hat{A}_{1t}y_{t-1} + \ldots + \hat{A}_{pt}y_{t-p} + \hat{C}_tD_t \]

the function returns a list for each equation with \( \hat{A}_{1t}, \ldots, \hat{A}_{pt}, \hat{C}_t \) set of arrays

Usage

tvAcoef(x)

## S3 method for class 'tvvar'
tvAcoef(x)
Arguments

\(x\) An object of class \texttt{tvvar} generated by \texttt{tvVAR}.

Value

A list object with coefficient arrays for the lagged endogenous variables.

Examples

\begin{verbatim}
data(Canada, package="vars")
 var.2p <- vars::VAR(Canada, p = 2, type = "const")
tvvar.2p <- tvVAR(Canada, p = 2, type = "const")
A <- vars::Acovf(var.2p)
tva <- tvAcovf(tvvar.2p)
\end{verbatim}

\texttt{tvVAR} \hspace{1cm} \textit{Time-Varying Autoregressive Model}

Description

\texttt{tvVAR} is used to fit an autorregressive model with time varying coefficients.

Usage

\begin{verbatim}
tvVAR(y, p = 1, z = NULL, bw = NULL, type = c("const", "none"),
 exogen = NULL, fixed = NULL, est = c("lc", "ll"), tkernel = c("Epa",
 "Gaussian"), singular.ok = TRUE)
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{y} A vector with the dependent variable.
\item \texttt{p} A scalar indicating the number of lags in the model.
\item \texttt{z} A vector with the smoothing variable.
\item \texttt{bw} An optional scalar or vector of length the number of equations. It represents the bandwidth in the estimation of coefficients. If NULL, it is selected by cross validation.
\item \texttt{type} A character 'const' if the model contains an intercept and 'none' otherwise.
\item \texttt{exogen} A matrix or data.frame with the exogenous variables (optional)
\item \texttt{fixed} (optional) numeric vector of the same length as the total number of parameters. If supplied, only NA entries in fixed will be varied.
\item \texttt{est} The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
\item \texttt{tkernel} The type of kernel used in the coefficients estimation method, one of Epanes- nikov ("Epa") or "Gaussian".
\item \texttt{singular.ok} Logical. If FALSE, a singular model is an error.
\end{itemize}
Details

It is a special case of linear model in which the regressors are lags of the dependent variable. If any variable is included in the \texttt{xreg} term, these are added to the regressors matrix. A time-varying coefficients linear regression (with an intercept if type = "const") is fitted.

Value

An object of class 'tvar' The object of class \texttt{tvar} have the following components:

- \texttt{tvcoef} A vector of dimension \texttt{obs} (\texttt{obs} = number of observations - number lags), with the time-varying coefficients estimates.
- \texttt{fitted} The fitted values.
- \texttt{residuals} Estimation residuals.
- \texttt{x} A matrix of model data, with lagged \texttt{y} and exogenous variables.
- \texttt{y} A vector with the dependent data used in the model.
- \texttt{z} A vector with the smoothing variable in the model.
- \texttt{bw} Bandwidth of mean estimation.
- \texttt{type} Whether the model has a constant or not.
- \texttt{exogen} A matrix or \texttt{data.frame} with other exogenous variables.
- \texttt{p} Number of lags
- \texttt{obs} Number of observations in estimation.
- \texttt{totobs} Number of observations in the original set.
- \texttt{level} Confidence interval range.
- \texttt{runs} Number of bootstrap replications.
- \texttt{tboot} Type of bootstrap.
- \texttt{BOOT} List with all bootstrap replications of \texttt{tvcoef}, if done.
- \texttt{call} Matched call.

References


See Also

\texttt{CI, plot}

\texttt{tvLM} for estimation of time-varying coefficients linear models, and \texttt{CI} for confidence intervals.
## Examples

```r
## Simulate an tvAR(2) process

tt <- (1:1000)/1000
beta <- cbind(0.5 * cos(2 * pi * tt), (tt - 0.5)^2)
y <- numeric(1000)
y[1] <- 0.5
y[2] <- -0.2

y(t) = beta1(t) y(t-1) + beta2(t) y(t-2) + ut

for (t in 3:1000)
{
  y[t] <- y[(t-1):(t-2)] %*% beta[t] + rnorm(1)
}
Y <- tail(y, 200)

## Estimate coefficients of process Y with ar.ols and tvAR
## and compare them in a plot

tvar.2p <- tvAR(Y, p = 2, type = "none", est = "ll", bw = 1.7)
AR.2p <- ar.ols(Y, aic = FALSE, order = 2, intercept = FALSE, demean = FALSE)
plot(tail(beta[, 1], 200), ylim=range(tvar.2p$tvcoef[, 1], tail(beta[, 1], 200)),
xlab = "", ylab = "", cex = 0.5, pch = 20)
abline(h = AR.2p$ar[1], col = 2)
lines(tvar.2p$tvcoef[, 1], col = 4)
legend("topleft", c(expression(beta[1]),"AR", "tvAR"), col = c(1, 2, 4),
  lty = 1, bty = "n")

## Estimate only coefficient from odd lags and the intercept

tvar.6p <- tvAR(Y, p = 6, type = "const", est = "ll")

## Generation of model with coefficients depending of a random variable

z <- arima.sim(n = 1000, list(ma = c(-0.2279, 0.2488)))
beta <- (z - 0.5)^2
y <- numeric(1000)
y[1] <- -1

y(t) = beta1(z(t)) y(t-1) + ut

for (t in 3:1000)
{
  y[t] <- y[(t-1)] %*% beta[t] + rnorm(1)
}
Y <- tail(y, 200)
Z <- tail(z, 200)

## Estimate coefficients of process Y with ar.ols and tvAR
## and compare them in a plot

tvar.2p.z <- tvAR(Y, z = Z, p = 1, type = "none", est = "ll")
```
tvBcoef

Coefficient Array of an Estimated tvVAR

Description

Returns the system estimated coefficients as an array.

Usage

```
tvBcoef(x)
```

## S3 method for class 'tvvar'
tvBcoef(x)

Arguments

x

An object of class 'tvvar', generated by `tvVAR`.

Details

Given an estimated time varying VAR of the form:

$$\tilde{y}_t = \hat{A}_{1t} y_{t-1} + \ldots + \hat{A}_{pt} y_{t-p} + \hat{C}_t D_t$$

the function returns a list for each equation with \((\hat{A}_{1t}| \ldots |\hat{A}_{pt}|\hat{C}_t)\) set of arrays.

Value

A list object with coefficient arrays for the lagged endogenous variables without including the intercept.
Examples

data(Canada, package="vars")
var.2p <- vars::VAR(Canada, p = 2, type = "const")
tvvar.2p <- tvVAR(Canada, p=2, type= "const")
B <- vars::Bcoef(var.2p)
tvB <- tvBcoef(tvvar.2p)

Description

Estimation of a time-varying variance-covariance matrix using the local constant or the local linear
kernel smoothing methodologies.

Usage

tvCov(x, bw, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"))

Arguments

x A matrix.
bw A scalar.
est A character, either "lc" or "ll" for local constant or local linear.
tkernel A character, either "Gaussian" or "Epa" kernel types.

Value

A matrix of dimension obs x neq x neq.

References


Examples

##Generate two independent (uncorrelated series)
y <- cbind(rnorm(100, sd = 4), rnorm(100, sd = 1))

##Estimation variance-variance matrix. If the bandwidth is unknown, it can
##calculated with function bwCov()
Sigma.hat <- tvCov(y, bw = 1.4)

##The first time estimate
print(Sigma.hat[1,])
##The mean over time of all estimates

### Generate two independent (uncorrelated series)
tvGLS

print(apply(Sigma.hat, 1:2, mean))
##Generate two dependent variables
y <- MASS::mvrnorm(n = 100, mu = c(0, 0), Sigma = cbind(c(1, -0.5), c(-0.5, 4)))

##Estimation variance-variance matrix
Sigma.hat <- tvCov(y, bw = 3.2)
##The first time estimate
print(Sigma.hat[,1])

tvGLS

**Time-varying Generalised Least Squares**

**Description**

**tvGLS** estimates time-varying coefficients of SURE using the kernel smoothing GLS.

**tvGLS** is used to estimate time-varying coefficients SURE using the kernel smoothing generalised least square.

**tvGLS** is used to estimate time-varying coefficients SURE using the kernel smoothing GLS

**Usage**

```r
tvGLS(x, ...)
```

## S3 method for class 'list'
```r
tvGLS(x, y, z = NULL, bw, Sigma = NULL, R = NULL, 
    r = NULL, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"), ...)
```

## S3 method for class 'tvsure'
```r
tvGLS(x, ...)
```

## S3 method for class 'matrix'
```r
tvGLS(x, y, z = NULL, bw, Sigma = NULL, R = NULL, 
    r = NULL, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"), ...)
```

**Arguments**

- **x** an object used to select a method.
- **y** A matrix.
- **z** A vector with the variable over which coefficients are smooth over.
- **bw** A numeric vector.
- **Sigma** An array.
- **R** A matrix.
- **r** A numeric vector.
- **est** Either "lc" or "ll".
- **tkernel** Either "Gaussian" or "Epa".
Details

The classical GLS estimator must be modified to generate a set of coefficients changing over time. The tvGls finds a GLS estimate at a given point in time \( t \) using the data near by. The size of the data window used is given by the bandwidth. The closest a point is to \( t \), the larger is its effect on the estimation which is given by the kernel. In this programme, the two possible kernels are the Epanechnikov and Gaussian. As in the classical GLS, the covariance matrix is involved in the estimation formula. If this matrix is NULL or the identity, then the programme returns the OLS estimates for time-varying coefficients.

Note, that unless with the tvSURE, the tvGLS may run with one common bandwidth for all equations or with a different bandwidths for each equation.

Value

tvGls returns a list containing:

- `tvcoef`: An array of dimension obs x nvar x neq (obs = number of observations, nvar = number of variables in each equation, neq = number of equations in the system) with the time-varying coefficients estimates.
- `fitted`: A matrix of dimension obs x neq with the fitted values from the estimation.
- `residuals`: A matrix of dimension obs x neq with the residuals from the estimation.

A list with the estimates, fitted and residuals values.

A list with the estimates, the fitted values and the residuals.

A list with the estimates, fitted and residuals values.

Examples

data(FF5F)
x <- list()
## SMALL/LoBM porfolios time-varying three factor model
x[[1]] <- FF5F[c("NA.Mkt.RF", "NA.SMB", "NA.HML", "NA.RMW", "NA.CMA")]
x[[2]] <- FF5F[c("JP.Mkt.RF", "JP.SMB", "JP.HML", "JP.RMW", "JP.CMA")]
x[[3]] <- FF5F[c("AP.Mkt.RF", "AP.SMB", "AP.HML", "AP.RMW", "AP.CMA")]
x[[4]] <- FF5F[c("EU.Mkt.RF", "EU.SMB", "EU.HML", "EU.RMW", "EU.CMA")]
y <- cbind(FF5F$NA.SMALL.LoBM, FF5F$JP.SMALL.LoBM, FF5F$AP.SMALL.LoBM, FF5F$EU.SMALL.LoBM)
##I fit the data with one bandwidth for each equation
ff5f.fit <- tvGls(x = x, y = y, bw = c(0.89, 1.55, 0.78, 0.31))

tvIRF

Time-Varying Impulse Response Function

Description

Computes the time-varying impulse response coefficients of an object of class tvvar, obtained with function tvVAR for n.ahead steps.
Usage

tvIRF(x, ...)

## S3 method for class 'tvvar'
tvIRF(x, impulse = NULL, response = NULL, n.ahead = 10,
     ortho = TRUE, ortho.cov = c("tv", "const"), bw.cov = NULL,
     cumulative = FALSE, ...)

Arguments

x       An object of class tvvar.
...
impulse A character vector of the impulses, default is all variables.
response A character vector of the responses, default is all variables.
n.ahead Integer specifying the steps.
ortho   Logical, if TRUE (the default) the orthogonalised IRF is computed.
ortho.cov A character indicating if the covariance matrix for the orthogonal tvIRF should be estimated as a constant or time varying. Either 'const' or 'tv' (default). This parameter is used only when ortho = TRUE.
bw.cov  A scalar (optional) with the bandwidth to estimate the errors variance-covariance matrix. If left NULL, it is estimated.
cumulative Logical, if TRUE the cumulated impulse response coefficients are computed. Default is FALSE.

Value

tvIRF returns and object of class tvirf with the following components:

irf An array dim = c(obs x number of response variables x number of impulse variables x number steps).
Lower An array with the lower confidence line, if calculated.
Upper An array with the upper confidence line, if calculated.
response A character, a number of a vector with the names or positions of the response(s) variable(s).
impulse A character, a number of a vector with the names or positions of the impulse(s) variable(s).
x A object of class tvvar

n.ahead Number of ahead impulse response functions.
ortho Logical, orthogonal or not impulse response function.
ortho.cov Character, either 'const' or 'tv' (default). This parameter is used when the orthogonal tv-IRF is calculated. The default is using an error time-varying variance-covariance.
bw.cov A scalar with the bandwidth to estimate the errors variance-covariance matrix. If NULL, it is calculated by cross-validation.
cumulative Logical, if TRUE the cumulated impulse response coefficients are computed. Default is FALSE.
level Numeric, confidence interval range. The default is zero.
runs Number of bootstrap replications.
tboot Type of bootstrap.
BOOT List with all bootstrap replications of tvirf, if done.

See Also
CI, plot

tvlM

Time-Varying Coefficients Linear Models

description
tvlM is used to fit a time-varying coefficients linear model

Usage
tvlM(formula, z = NULL, data, bw = NULL, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"), singular.ok = TRUE)

Arguments
formula An object of class formula.
z A vector with the smoothing variable.
data An optional data frame or matrix.
bw An optional scalar. It represents the bandwidth in the estimation of trend coefficients. If NULL, it is selected by cross validation.
est The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
tkernel The type of kernel used in the coefficients estimation method, one of Epanesnikov ("Epa") or "Gaussian".
singular.ok Logical. If FALSE, a singular model is an error.
Details

Models for \texttt{tvLM} are specified symbolically using the same formula format than function \texttt{lm}. A typical model has the form $\text{response} \sim \text{terms}$ where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. A specification of the form first:second indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

A formula has an implied intercept term. To remove this use either $y \sim x - 1$ or $y \sim 0 + x$.

Value

An object of class \texttt{tvLM} The object of class \texttt{tvLM} have the following components:

- tvcoef: A matrix of dimensions
- fitted: The fitted values.
- residuals: Estimation residuals.
- x: A matrix with the regressors data.
- y: A vector with the dependent variable data.
- z: A vector with the smoothing variable.
- bw: Bandwidth of mean estimation.
- est: Nonparametric estimation methodology.
- tkernel: Kernel used in estimation.
- level: Confidence interval range.
- runs: Number of bootstrap replications.
- tboot: Type of bootstrap.
- boot: List with all bootstrap replications of tvcoef, if done.
- call: Matched call.

References


See Also

- CI, \texttt{plot}
- \texttt{bw} for bandwidth selection, \texttt{tvOLS} for the estimation procedure and \texttt{CI} for confidence intervals.
Examples

```r
# Simulate a linear process with time-varying coefficient
# as functions of scaled time.

tau <- seq(0, 1, length.out = 200)
beta <- data.frame(betal = sin(2 * pi * tau), beta2 = 2 * tau)
X1 <- rnorm(200)
X2 <- rchisq(200, df = 4)
error <- rt(200, df = 10)
y <- apply(cbind(X1, X2) * beta, 1, sum) + error
data <- data.frame(y = y, X1 = X1, X2 = X2)

# Estimate coefficients with lm and tvlm for comparison

coef.lm <- stats::lm(y ~ 0 + X1 + X2, data = data)$coef
model.tvlm <- tvlm(y ~ 0 + X1 + X2, data = data, bw = 0.2)
```

---

**tvOLS**

*Time-Varying Ordinary Least Squares*

**Description**

tvOLS is used to fit univariate linear models with time-varying coefficients.

**Usage**

```r
tvOLS(x, y, z = NULL, bw, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"), singular.ok = singular.ok)
```

**Arguments**

- `x`: A matrix with all regressors.
- `y`: A vector with dependent variable.
- `z`: A vector with the variable over which coefficients are smooth over.
- `bw`: A numeric vector.
- `est`: The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
- `tkernel`: The type of kernel used in the coefficients estimation method, one of Epanechnikov ("Epa") or "Gaussian".
- `singular.ok`: Logical. If FALSE, a singular model is an error.

**Value**

A list with the estimates, fitted and residuals values.
See Also

bw for bandwidth selection and tvLM

Examples

tau <- seq(1:500)/500
beta <- data.frame(beta1 = sin(2*pi*tau), beta2 = 2*tau)
X <- data.frame(X1 = rnorm(500), X2 = rchisq(500, df = 4))
error <- rt(500, df = 10)
y <- apply(X*beta, 1, sum) + error
coeff.lm <- stats::lm(y~0+X1+X2, data = X)$coef
coeff.tvlm <- tvOLS(x = X, y = y, bw = 0.1)$tvcoef
plot(tau,beta[, 1], type="l", main="", ylab = expression(beta[1]), xlab = expression(tau),
ylim = range(beta[,1], coef.tvlm[, 1]))
abline(h = coef.lm[1], col = 2)
lines(tau, coef.tvlm[, 1], col = 4)
legend("topright", c(expression(beta[1]), "lm", "tvlm"), col = c(1, 2, 4), bty="n", lty = 1)

Description

Returns the estimated time-varying coefficient arrays of the moving average representation of a stable tvvar object obtained with function tvVAR.

Usage

tvPhi(x, nstep = 10, ...)

## S3 method for class 'tvvar'
tvPhi(x, nstep = 10, ...)

Arguments

x An object of class tvvar.

nstep An integer specifying the number of moving error coefficient matrices to be calculated.

... Other parameters passed to specific methods.
**tvPsi**

*Time-Varying Coefficient Arrays of the Orthogonalised MA Representation*

**Description**

Returns the estimated orthogonalised time-varying coefficient arrays of the moving average representation of a stable tvvar object obtained with function tvVAR.

**Usage**

```r
tvPsi(x, nstep = 10, ...) 
```

## S3 method for class 'tvvar'

```
tvPsi(x, nstep = 10, ortho.cov = "const", bw.cov = NULL, 
      ...)  
```

**Arguments**

- `x` An object of class tvvar, generated by tvVAR().
- `nstep` An integer specifying the number of orthogonalised moving error coefficient matrices to be calculated for each time t.
- `...` Other parameters passed to specific methods.
- `ortho.cov` A character either 'const' if the error cov matrix must be estimated by a constant or 'tv' if it is estimated as a time-varying matrix. Default is 'const'.
- `bw.cov` A scalar (optional) with the bandwidth to estimate the errors variance-covariance matrix.

**Value**

A list with an array of dimensions (obs x neq x neq nstep + 1) holding the estimated time varying coefficients of the moving average representation, and the bandwidth used to estimate the covariance matrix (optional).

---

**tvSURE**

*Time-Varying Seemingly Unrelated Regression Equations Model*

**Description**

Fits a set of balanced linear structural equations using Time-varying Ordinary Least Squares (tvOLS), Time-varying Seemingly Unrelated Regression (tvGLS), when the error variance-covariance matrix is known, or Time-varying Feasible Seemingly Unrelated Regression (tvFGLS), when the error variance-covariance matrix is unknown.
Usage

tvSURE(formula, z = NULL, bw = NULL, data, method = c("tvOLS", "tvFGLS", "tvGLS"), Sigma = NULL, est = c("lc", "ll"), tkernel = c("Epa", "Gaussian"), bw.cov = NULL, singular.ok = TRUE, R = NULL, r = NULL, control = tvsure.control(...), ...)

Arguments

- **formula**: A list of formulas, one for each equation.
- **z**: A vector containing the smoothing variable.
- **bw**: An optional scalar or vector of length the number of equations. It represents the bandwidth in the estimation of trend coefficients. If NULL, it is selected by cross validation.
- **data**: A matrix or data frame containing variables in the formula.
- **method**: A character, a matrix of dimensions neq x neq or an array of dimensions obs x neq x neq, where obs is the number of observations and neq is the number of equations. If method = identity or tvOLS (default) then the method used is a time-varying OLS. If method is a matrix (constant over time) or an array, then the tvGLS is called. If method = tvFGLS, then the covariance matrix is estimated nonparametrically and the estimation of the system is done as a whole.
- **Sigma**: A matrix of dimensions neq x neq or an array of dimensions neq x neq x obs (neq = number of equations, obs = number of observations). It represents the covariance matrix of the error term. Only necessary for method tvGLS.
- **est**: The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
- **tkernel**: The type of kernel used in the coefficients estimation method, one of Epanesnikov ("Epa") or "Gaussian".
- **bw.cov**: An optional scalar. It represents the bandwidth in the "lc" nonparametric estimation of the time-varying covariance matrix. If NULL, it is selected by cross validation.
- **singular.ok**: Logical. If FALSE, a singular model is an error.
- **R**: An optional nrest x nvar x neq (nrest = number of restrictions, nvar = number of variables in each equation, neq = number of equations).
- **r**: An optional vector of length the number of restrictions. By default it contains zeros.
- **control**: list of control parameters. The default is constructed by the function tvsure.control. See the documentation of tvsure.control for details.
- **...**: Other parameters passed to specific methods.

Details

This function wraps up the kernel smoothing "tvOLS" and "tvGLS" estimators. The former is used when equations are considered independent while the later assumes that the error term is correlated amongst equations. This relation is given in matrix "Sigma" which is used in the estimation.
When "Sigma" is known, the estimates are calculated via the "tvGLS", and via the "tvFGLS" when "Sigma" is unknown and must be estimated.

Bandwidth selection is of great importance in kernel smoothing methodologies and it is done automatically by cross-validation. One important aspect in the current packages is that the bandwidth is selected independently for each equation and then the average is taken to use the same bandwidth for each equation. It has been shown in Casas et al. (2017) that using different bandwidths for each equation is in general a bad practice, even for uncorrelated equations. Even though, the user may be able to use different bandwidths calling functions \texttt{bw} and \texttt{tvGLS} separately.

A system consists of "neq" number of equations with "obs" number of observations each and a number of variables not necessarily equal for all equations. The matrix notation is:

\[
Y_t = X_t \beta_t + u_t
\]

where \( Y_t = (y_{1t}, y_{2t}, \ldots, y_{neqt})' \), \( X_t = diag(x_{1t}, x_{2t}, \ldots, x_{neqt}) \) and \( \beta_t = (\beta_{1t}', \ldots, \beta_{neqt}')' \) is a vector of order the total number of variables in the system. The error vector \( u_t = (u_{1t}, u_{2t}, \ldots, u_{neqt})' \) has zero mean and covariance matrix \( E(u_t u_t') = \Sigma_t \).

\textbf{Value}

tvSURE returns a list of the class \texttt{tvsure} containing the results of the whole system, results of the estimation and confidence intervals if chosen. The object of class \texttt{tvsure} have the following components:

- \texttt{tvcoef} An array of dimension \( \text{obs x nvar x neq} \) (\( \text{obs} = \text{number of observations}, \text{nvar} = \text{number of variables in each equation}, \text{neq} = \text{number of equations in the system} \)) with the time-varying coefficients estimates.
- \texttt{Lower} If \texttt{level} non equal zero, an array of dimension \( \text{obs x nvar x neq} \) containing the confidence interval lower band.
- \texttt{Upper} If \texttt{level} non equal zero, an array of dimension \( \text{obs x nvar x neq} \) containing the confidence interval upper band.
- \texttt{Sigma} An array of dimension \( \text{obs x neq x neq} \) with the estimates of the errors covariance matrix.
- \texttt{fitted} The fitted values.
- \texttt{residuals} Estimation residuals.
- \texttt{x} A list with the regressors data.
- \texttt{y} A matrix with the dependent variable data.
- \texttt{z} A vector with the smoothing variable.
- \texttt{bw} Bandwidth of mean estimation.
- \texttt{obs} Integer specifying the number of observations in each equation (balanced sample).
- \texttt{neq} Integer specifying the number of equations.
- \texttt{nvar} Vector of integers specifying the number of variables in each equation.
- \texttt{method} Estimation method.
- \texttt{est} Nonparemtric estimation methodology.
tvVAR

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tkernel</td>
<td>Kernel type.</td>
</tr>
<tr>
<td>bw.cov</td>
<td>Bandwidth of Sigma estimation.</td>
</tr>
<tr>
<td>level</td>
<td>Confidence interval range.</td>
</tr>
<tr>
<td>runs</td>
<td>Number of bootstrap replications.</td>
</tr>
<tr>
<td>tboot</td>
<td>Type of bootstrap.</td>
</tr>
<tr>
<td>BOOT</td>
<td>List with all bootstrap replications of tvcoef, if done.</td>
</tr>
<tr>
<td>R</td>
<td>Restrictions matrix.</td>
</tr>
<tr>
<td>r</td>
<td>Restrictions vector.</td>
</tr>
<tr>
<td>formula</td>
<td>Initial formula.</td>
</tr>
<tr>
<td>call</td>
<td>Matched call.</td>
</tr>
</tbody>
</table>

References


See Also

- *systemfit*
- CI, plot
  - *bw* for bandwidth calculation, *tvGLS* for nonparametric estimator and *CI* for confidence intervals.

---

**Description**

Fits a time-varying coefficients vector autorregressive model with p lags.

**Usage**

```r
tvVAR(y, p = 1, z = NULL, bw = NULL, type = c("const", "none"), exogen = NULL, est = c("1c", "1l"), tkernel = c("Epa", "Gaussian"), singular.ok = TRUE)
```

---
Arguments

y A matrix with dimension obs x neq (obs = number of observations and neq = number of equations)

p A scalar indicating the number of lags in the model

z A vector containing the smoothing variable.

bw An optional scalar or vector of length the number of equations. It represents the bandwidth in the estimation of trend coefficients. If NULL, it is selected by cross validation.

type A character ‘const’ if the model contains an intercept and ‘none’ otherwise.

exogen A matrix or data.frame with the exogenous variables (optional)

est The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.

tkernel The type of kernel used in the coefficients estimation method, one of Epanechnikov ("Epa") or "Gaussian".

singular.ok Logical. If FALSE, a singular model is an error.

Value

An object of class ‘tvvar’ The object of class tvvar have the following components:

tvcoef An array of dimension obs x neq (obs = number of observations, neq = number of equations in the system) with the time-varying coefficients estimates.

fitted The fitted values.

residuals Estimation residuals.

x A list with the regressors data and the dependent variable.

y A matrix with the dependent variable data.

bw Bandwidth of mean estimation.

type Whether the model has a constant or not.

exogen A matrix or data.frame with other exogenous variables.

p Number of lags

neq Number of equations

obs Number of observations in estimation.

totobs Number of observations in the original set.

call Matched call.

See Also

CI, plot
Examples

```r
## Inflation rate, unemployment rate and treasury bill interest rate for the US,
## as used by Primiceri (2005).
data(usmacro, package = "bvarsv")

model.VAR <- vars::VAR(usmacro, p = 6, type = "const")
model.tvVAR <- tvVAR(usmacro, p = 6, type = "const", bw = c(1.8, 20, 20))
plot(model.tvVAR)
```

---

**Description**

Update and Re-fit the Models of package tvReg

**Usage**

```r
## S3 method for class 'tvsure'
update(object, y = NULL, ...)

## S3 method for class 'tvlm'
update(object, y = NULL, ...)

## S3 method for class 'tvar'
update(object, y = NULL, ...)

## S3 method for class 'tvvar'
update(object, y = NULL, ...)
```

**Arguments**

- `object` An object used to select a method.
- `y` The dependent variable to update the model.
- `...` Other parameters passed to specific methods.

**Value**

An object of class tvsure.
An object of class tvlm.
An object of class tvar.
An object of class tvvar.
Index

+Topic coefficients, tvAR, 10
+Topic coefficients
tvlM, 18
+Topic datasets
CEES, 4
FF5F, 6
+Topic linear
tvlM, 18
+Topic models,
tvlM, 18
tvSURE, 22
+Topic nonparametric
tvAR, 10
tvlM, 18
tvSURE, 22
+Topic regression,
tvlM, 18
+Topic regression
tvlM, 18
tvSURE, 22
+Topic statistics
tvAR, 10
tvlM, 18
tvSURE, 22
+Topic time-varying
tvAR, 10
tvlM, 18
+Topic time
tvlM, 18
tvSURE, 22
+Topic varying
tvlM, 18
tvSURE, 22

bw, 2, 19, 21, 24, 25
bwCov, 3

CEES, 4
CI, 5, 11, 18, 19, 25, 26

FF5F, 6
plot, 11, 18, 19, 25, 26
plot.tvar (plot.tvsure), 8
plot.tvirf (plot.tvsure), 8
plot.tvlm (plot.tvsure), 8
plot.tvvar (plot.tvsure), 8
systemfit, 25
tvAcoef, 9
tvAR, 6, 9, 10
tvar (tvAR), 10
tvar-class (tvAR), 10
tvBcoef, 13
tvCov, 14
tvGLS, 15, 24, 25
tvIRF, 16
tvIRF-class (tvIRF), 16
tvIRF (tvIRF), 16
tvIRF.tvvar (tvIRF), 16
tvlM, 6, 9, 11, 18, 21
tvlm (tvLM), 18
tvlm-class (tvLM), 18
tvOLS, 19, 20
tvPhi, 21
tvPsi, 22
tvSURE, 6, 9, 22
tvsure (tvSURE), 22
tvsure-class (tvSURE), 22
tvsure.control, 23
tvVAR, 6, 9, 10, 13, 25
tvvar (tvVAR), 25
tvvar-class (tvVAR), 25
update.tvar (update.tvsure), 27
update.tvlm (update.tvsure), 27
update.tvsure, 27
update.tvvar (update.tvsure), 27

28