Package ‘tvReg’

March 16, 2026

Type Package

Title Time-Varying Coefficient for Single and Multi-Equation Regressions

Version 0.5.11

Date 2026-03-12

Description Fitting time-varying coefficient models for single and multi-equation regressions, using kernel smoothing techniques.

License GPL (>= 3)

LazyData yes

Depends R (>= 3.6), Matrix, graphics, stats (>= 2.14.0), methods

Imports systemfit (>= 1.1-20), MASS, vars, bvarsv, plm

Suggests knitr, rmarkdown

URL `https://github.com/icasas/tvReg`

BugReports `https://github.com/icasas/tvReg/issues`

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

Author Isabel Casas [aut, cre],
Ruben Fernandez-Casal [aut]

Maintainer Isabel Casas <`casasis@gmail.com`>

Repository CRAN

NeedsCompilation no

Date/Publication 2026-03-16 08:10:10 UTC

Contents

tvReg-package 2
bw 3
bwCov 5

CEES 6
confint.tvlm 7
FF5F 10
forecast 11
OECD 13
plot.tvsure 14
predict.tvlm 15
print.tvlm 17
RV 18
summary.tvlm 19
tvAcoef 20
tvAR 20
tvBcoef 23
tvCor 24
tvCov 25
tvFE 26
tvGLS 28
tvIRF 30
tvLM 32
tvOLS 34
tvPhi 36
tvPLM 36
tvPsi 39
tvRE 40
tvSURE 41
tvVAR 45
update.tvlm 47

Index **48**

`tvReg-package` *tvReg: Time-Varying Coefficient for Single and Multi-Equation Regressions*

Description

This package covers a large range of semiparametric regression methods with time-varying coefficients using nonparametric kernel smoothing for the estimation.

Main functions

The five basic functions in this package are `tvLM`, `tvAR`, `tvSURE`, `tvPLM`, `tvVAR` and `tvIRF`. Moreover, this package provides the `confint`, `fitted`, `forecast`, `plot`, `predict`, `print`, `resid` and `summary` methods adapted to the class attributes of the `tvReg`. In addition, it includes bandwidth selection methods, time-varying variance-covariance estimators and four estimation procedures: the time-varying ordinary least squares, which are implemented in the `tvOLS` methods, the time-varying generalised least squares for a list of equations, which is implemented in the `tvGLS` methods, time-varying pooled and random effects estimators for panel data, which are implemented in the `tvRE` and the time-varying fixed effects estimator, which is implemente in the `tvFE`.

Further information

Details on the theory and applications to finance and macroeconomics can be found in Casas and Fernandez-Casal (2022, `https://journal.r-project.org/articles/RJ-2022-002/`), and in the package vignette `https://icasas.github.io/tvReg/articles/tvReg.html`.

Acknowledgments

Funded by the Horizon 2020. Framework Programme of the European Union.

Author(s)

Isabel Casas (<`casasis@gmail.com`>), Ruben Fernandez-Casal (<`rubenfcasal@gmail.com`>).

References

Casas, I. and Fernandez-Casal, R., *tvReg: Time-varying Coefficient Linear Regression for Single and Multi-Equations in R* (June, 2022). The R Journal. Available at `https://journal.r-project.org/articles/RJ-2022-002/`.

---

bw — *Bandwidth Selection by Cross-Validation*

Description

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

Usage

```
bw(x, ...)

## Default S3 method:
bw(
  x,
  y,
  z = NULL,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  singular.ok = TRUE,
  ...
)

## S3 method for class 'list'
bw(
  x,
  y,
  z = NULL,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  singular.ok = TRUE,
  ...
)

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

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

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

## S3 method for class 'tvsure'
bw(x, ...)

## S3 method for class 'tvplm'
bw(x, ...)

## S3 method for class 'pdata.frame'
bw(
  x,
  z = NULL,
  method,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  ...
)
```

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.
`cv.block`	A positive scalar with the size of the block in leave-one block-out cross-validation. By default 'cv.block=0' meaning leave-one-out cross-validation.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`singular.ok`	Logical. If FALSE, a singular model is an error.
`method`	A character with the choice of panel model/estimation method:

- `"tvPOLS"` (default): the data is pooled and estimated with time-varying OLS. No individual or time effects are estimated.
- `"tvFE"`: individual effects which might be correlated with the regressors are estimated.
- `"tvRE"`: individual effects are considered random and independent of the regressors.

Value

A scalar or a vector of scalars with the bandwidth(s) selected by cross-validation.

Examples

```
##Generate data
tau <- seq(1:200)/200
beta <- 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

##Select bandwidth by cross-validation
bw <- bw(X, y, est = "ll", tkernel = "Gaussian")

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)

## Select bandwidth by cross-validation
bw <- bw(x = x, y = y)

##One bandwidth per equation
print(bw)
```

---

bwCov *Covariance Bandwidth Calculation by Cross-Validation*

Description

bwCov calculates a single bandwidth to estimate the time-varying variance- covariance matrix.

Usage

```
bwCov(
  x,
  z = NULL,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian")
)
```

Arguments

`x`	A matrix or a data frame.
`z`	A vector with the variable over which coefficients are smooth over.
`cv.block`	A positive scalar with the size of the block in leave-one block-out cross-validation. By default 'cv.block=0' meaning leave-one-out cross-validation.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.

Value

A scalar.

Examples

```
data(CEES)
## Using a shorter set for a quick example. Variable "Date" is removed.
mydata <- tail (CEES[, -1], 50)
bw.cov <- bwCov(mydata)
Sigma.hat <- tvCov(mydata, bw = bw.cov)
```

CEES *Standarised rates from a currency portfolio.*

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 standarised rates after "devolatilisation", i.e. standarising the rates using a GARCH(1,1) estimate of the volatility.

Format

A data frame with 2855 rows and 8 variables. Below the standarised rates of daily US dollar exchange rates of

Date Daily data from Jan 6, 1999 until May 7, 2010 - without weekends and days off

AUS Australian dollar

CHF Swiss franc

EUR Euro

GBP British pound

RAND South African rand

REALB Brazilian real

YEN Japanese yen

References

Aslanidis, N. and Casas, I. (2013) Nonparametric correlation models for portfolio allocation, *Journal of Banking and Finance*, 37, 2268 - 2283.

`confint.tvlm` *Confidence Intervals for Objects in tvReg*

Description

confint is used to estimate the bootstrap confidence intervals for objects with class attribute `tvlm`, `tvar`, `tvirf`, `tvsure` and `tvplm`.

Usage

```
## S3 method for class 'tvlm'
confint(
  object,
  parm,
  level = 0.95,
  runs = 100,
  tboot = c("wild", "wild2"),
  ...
)

## S3 method for class 'tvar'
confint(
  object,
  parm,
  level = 0.95,
  runs = 100,
  tboot = c("wild", "wild2"),
  ...
)

## S3 method for class 'tvsure'
confint(
  object,
  parm,
  level = 0.95,
  runs = 100,
  tboot = c("wild", "wild2"),
  ...
)

## S3 method for class 'tvvar'
confint(
  object,
  parm,
  level = 0.95,
  runs = 100,
  tboot = c("wild", "wild2"),
  ...
)

## S3 method for class 'tvirf'
confint(
  object,
  parm,
  level = 0.95,
  runs = 100,
  tboot = c("wild", "wild2"),
  ...
)

## S3 method for class 'tvplm'
confint(
  object,
  parm,
  level = 0.95,
  runs = 100,
  tboot = c("wild", "wild2"),
  ...
)
```

Arguments

`object`	An object used to select a method.
`parm`	A specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are consid-

	ered.
`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.
`...`	Other parameters passed to specific methods.

Value

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

References

Chen, X. B., Gao, J., Li, D., and Silvapulle, P (2017) Nonparametric estimation and forecasting for time-varying coefficient realized volatility models, *Journal of Business and Economic Statistics*, 36, 88-100.

Mammen, E (1993) Bootstrap and wild bootstrap for high dimensional linear models, *Annals of Statistics*, 21, 255-285.

See Also

`tvLM`, `tvAR`, `tvVAR`, `tvSURE`

Examples

```
## Not run:
##Calculation of confidence intervals for a TVLM model

##Generation of time-varying coefficients linear model
set.seed(42)
tau <- seq(1:200)/200
beta <- data.frame(beta1 = 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)

##Fitting the model and confidence interval calculation
model.tvlm <-  tvLM(y ~ 0 + X1 + X2, data = data, bw = 0.29)
tvci <- confint(model.tvlm, level = 0.95, runs = 20)

##If a second confidence interval on the "same" object is calculated,
##for example with a different level, the calculation is faster
tvci.80 <- confint(tvci, level = 0.8)

## End(Not run)
```

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

Description

A dataset containing the 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 markets: North America (NA), Japan (JP), Asia Pacific (AP) 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 Monthly returns of portfolio SMALL/LoBM in North American market

NA.SMALL.HiBM Monthly returns of portfolio SMALL/HiBM in North American market

NA.BIG.LoBM Monthly returns of portfolio BIG/LoBM in North American market

NA.BIG.HiBM Monthly returns of portfolio BIG/HiBM in North American market

NA.Mkt.RF North American market excess returns, i.e return of the market - market risk free rate

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 Monthly returns of portfolio SMALL/LoBM in Japanese market

JP.SMALL.HiBM Monthly returns of portfolio SMALL/HiBM in Japanese market

JP.BIG.LoBM Monthly returns of portfolio BIG/LoBM in Japanese market

JP.BIG.HiBM Monthly returns of portfolio BIG/HiBM in Japanese market

JP.Mkt.RF Japanese market excess returns, i.e return of the market - market risk free rate

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 Monthly returns of portfolio SMALL/LoBM in Asia Pacific market

AP.SMALL.HiBM Monthly returns of portfolio SMALL/HiBM in Asia Pacific market

AP.BIG.LoBM Monthly returns of portfolio BIG/LoBM in Asia Pacific market

AP.BIG.HiBM Monthly returns of portfolio BIG/HiBM in Asia Pacific market

AP.Mkt.RF Asia Pacific market excess returns, i.e return of the market - maket risk free rate

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 returns, i.e returns of the market - market risk free rate

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

Kennet R. French - Data Library (2017) http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html#Internatio

Fama, E. and French, K. R (1993) Common risk factors in the returns on stocks and bonds, *Journal of Financial Economics*, 3-56.

Fama, E. F. and French, K. R (2015) A five-factor asset pricing model, *Journal of Financial Economics*, 116, 1-22.

`forecast` *Forecast Methods for Objects in tvReg.*

Description

`forecast` calculates the forecast for objects with class attribute `tvlm`, `tvar`, `tvvar`, `tvirf`, `tvsure` and `tvplm`. If the smoothing variable (z) in the model is non-NULL and it is a random variable then use function `predict` with parameter `newz`.

Usage

```
forecast(object, ...)

## S3 method for class 'tvlm'
forecast(object, newdata, n.ahead = 1, winsize = 0, ...)

## S3 method for class 'tvar'
forecast(object, n.ahead = 1, newz = NULL, newexogen = NULL, winsize = 0, ...)

## S3 method for class 'tvvar'
forecast(object, n.ahead = 1, newz = NULL, newexogen = NULL, winsize = 0, ...)

## S3 method for class 'tvsure'
forecast(object, newdata, n.ahead = 1, winsize = 0, ...)

## S3 method for class 'tvplm'
forecast(object, newdata, n.ahead = 1, winsize = 0, ...)
```

Arguments

`object`	An object used to select a method.
`...`	Other parameters passed to specific methods.
`newdata`	A matrix or data.frame with the values of the regressors to use for forecasting.
`n.ahead`	A scalar with the forecast horizon, value 1 by default.
`winsize`	A scalar. If 0 then an 'increase window' forecasting is performed. Otherwise a 'rolling window' forecasting is performed with window size given by 'winsize'.
`newz`	A vector with the new values of the smoothing variable.
`newexogen`	A matrix or vector with the new values of the exogenous variables. Only for predictions of *tvar* and *tvvar* objects.

Value

An object of class matrix or vector with the same dimensions than the dependent variable of `object`.

See Also

`predict`.

Examples

```
data("RV")
RV2 <- head(RV, 2001)
TVHAR <- tvLM (RV ~ RV_lag + RV_week + RV_month, data = RV2, bw = 20)
newdata <- cbind(RV$RV_lag[2002:2004], RV$RV_week[2002:2004],
              RV$RV_month[2002:2004])
forecast(TVHAR, newdata, n.ahead = 3)
```

```
data("RV")
exogen = RV[1:2001, c("RV_week", "RV_month")]
TVHAR2 <- tvAR(RV$RV_lag[1:2001], p = 1, exogen = exogen, bw = 20)
newexogen <- RV[2002:2004, c("RV_week", "RV_month")]
forecast(TVHAR2, n.ahead = 3, newexogen = newexogen)

data(usmacro, package = "bvarsv")
tvVAR.fit <- tvVAR(usmacro, p = 6, type = "const", bw = c(1.8, 20, 20))
forecast(tvVAR.fit, n.ahead = 10)

data("Kmenta", package = "systemfit")
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice
system <- list(demand = eqDemand, supply = eqSupply)
tvOLS.fit <- tvSURE(system, data = Kmenta, est = "ll", bw = c(1.5, 1.5))
newdata <- data.frame(price = c(90, 100, 103), farmPrice = c(70, 95, 103),
income = c(82, 94, 115))
forecast(tvOLS.fit, newdata = newdata, n.ahead = 3)
data(OECD)
tvpols <- tvPLM(lhe~lgdp+pop65+pop14+public, index = c("country", "year"),
data = OECD, method = "pooling", bw =  8.9)
newdata <- OECD[c(7, 9), 4:7]
forecast(tvpols, newdata = newdata, n.ahead = 2)
```

`OECD` *Variables related to the problem of healthcare spending.*

Description

Variables related to the problem of healthcare spending.

Format

A data frame with 680 rows and 7 columns.

country

year

lhe Log of country's healthcare spending

lgdp log of country's gdp

pop65 Country's ratio of population greater than 65 years old

pop14 Country's ratio of population younger than 15 years old

public Country's ratio of healthcare funding coming from the government

References

Casas, I., Gao, J., Peng, B. and Xie, S. (2021). Time-Varying Income Elasticities of Healthcare Expenditure for the OECD and Eurozone. *Journal of Applied Econometrics, 36, pp. 328-345.*

`plot.tvsure` *Plot Methods for Objects in tvReg*

Description

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

Usage

```
## S3 method for class 'tvsure'
plot(x, eqs = NULL, vars = NULL, plot.type = c("multiple", "single"), ...)

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

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

## S3 method for class 'tvplm'
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"),
  ...
)
```

Arguments

`x`	An object used to select the method.
`eqs`	A vector of integers. Equation(s) number(s) of the coefficients to be plotted.
`vars`	A vector of integers. Variable number(s) of the coefficients to be plotted.
`plot.type`	Character, if multiple all plots are drawn in a single device, otherwise the plots are shown consecutively.
`...`	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.

See Also

`tvLM`, `tvAR`, `tvVAR`, `tvSURE`, `tvPLM`

`predict.tvlm` *Predict Methods for Objects in tvReg.*

Description

Predict methods for objects with class attribute `tvlm`, `tvar`, `tvvar`, `tvirf`, `tvsure` and `tvplm`. This function needs new values of variables y (response), x (regressors), exogen (exogenous variables, when used), and z (smoothing variable).

Usage

```
## S3 method for class 'tvlm'
predict(object, newdata, newz, ...)

## S3 method for class 'tvar'
predict(object, newdata, newz, newexogen = NULL, ...)

## S3 method for class 'tvvar'
predict(object, newdata, newz, newexogen = NULL, ...)

## S3 method for class 'tvsure'
predict(object, newdata, newz, ...)

## S3 method for class 'tvplm'
predict(object, newdata, newz, ...)
```

Arguments

`object`	An object used to select a method.
`newdata`	A pdata.frame with new values of all regressors, with the same name and order as they appear in argument 'data' from the 'tvplm' object
`newz`	A vector with new values of the smoothing variable.
`...`	Other arguments passed to specific methods.
`newexogen`	A matrix or vector with the new value of the exogenous variables. Only for predictions of 'tvar' and 'tvvar' objects.

Value

An object of class matrix or vector with the prediction.

See Also

forecast.

Examples

```
## Example of TVLM prediction with coefficients as
## functions of the realized quarticity

data("RV")
RV2 <- head(RV, 2001)
z <- RV2$RQ_lag_sqrt
TVHARQ <- tvLM (RV ~ RV_lag + RV_week + RV_month,
                 z = z, data = RV2, bw = 0.0062)
newdata <- cbind(RV$RV_lag[2002:2004], RV$RV_week[2002:2004],
              RV$RV_month[2002:2004])
newz <- RV$RQ_lag_sqrt[2002:2004]
predict(TVHARQ, newdata, newz)

## Example of TVAR prediction with coefficients as
## functions of the realized quarticity
data("RV")
RV2 <- head(RV, 2001)
exogen = RV2[, c("RV_week", "RV_month")]
TVHARQ2 <- tvAR (RV2$RV, p = 1, exogen = exogen,
                      z = RV2[, "RQ_lag_sqrt"], bw = 0.0062)
newylag <- RV$RV[2002:2004]
newz <- RV$RQ_lag_sqrt[2002:2004]
newexogen <- RV[2002:2004, c("RV_week", "RV_month")]
predict(TVHARQ2, newylag,  newz, newexogen = newexogen)
## Example of TVVAR prediction with coefficients as
## functions of a random ARMA (2,2) process

data(usmacro, package = "bvarsv")
smoothing <- arima.sim(n = NROW(usmacro) + 3,
list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)),
sd = sqrt(0.1796))
smoothing <- as.numeric(smoothing)
TVVAR.z <- tvVAR(usmacro, p = 6, type = "const",
               z = smoothing[1:NROW(usmacro)], bw = c(16.3, 16.3, 16.3))
newdata <- data.frame(inf = c(2, 1, 6), une = c(5, 4, 9), tbi = c(1, 2.5, 3))
newz <- c(0, 1.2, -0.2)
predict(TVVAR.z, newdata = newdata, newz = newz)

## Example of TVSURE prediction with coefficients as
## functions of an ARMA(2,2) process
data("Kmenta", package = "systemfit")
nobs <- NROW (Kmenta)
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice
system <- list(demand = eqDemand, supply = eqSupply)
smoothing <- arima.sim(n = nobs + 3,
                       list(ar = c(0.8897, -0.4858), ma = c(-0.2279, 0.2488)),
                       sd = sqrt(0.1796))
smoothing <- as.numeric(smoothing)
TVOLS.z <- tvSURE(system, data = Kmenta,
                      z = smoothing[1:nobs],  bw = c(7, 1.8),
                      est = "ll")
newdata <- data.frame(consump = c(95, 100, 102), price = c(90, 100, 103),
                      farmPrice = c(70, 95, 103), income = c(82, 94, 115))
newz <- tail(smoothing, 3)
predict(TVOLS.z, newdata = newdata, newz = newz)

data(OECD)
z <- runif(length(levels(OECD$year)), 10, 15)
TVPOLS <- tvPLM(lhe~lgdp+pop65+pop14+public, z = z,
index = c("country", "year"), data = OECD,  method ="pooling", bw =  2)
newdata <- cbind(lgdp = c(10, 13), pop65 = c(9, 12),
pop14 = c(17, 30), public = c(13, 20))
newz <- runif(2, 10, 15)
predict(TVPOLS, newdata = newdata, newz = newz)
```

print.tvlm *Print results of functions in tvReg*

Description

Print some results for objects with class attribute `tvlm`, `tvar`, `tvvar`, `tvirf`, `tvsure` and `tvplm`.

Usage

```
## S3 method for class 'tvlm'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvar'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvplm'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvsure'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvvar'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvirf'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	An object used to select a method.
`digits`	An integer, indicating the minimal number of significant digits.
`...`	Other parameters passed to specific methods.

Details

These functions print a few results from the time-varying estimated coefficients

See Also

`plot.tvlm`, `plot.tvar`, `plot.tvvar`, `plot.tvirf`,`plot.tvsure`, `plot.tvplm`

`RV` *Daily realized variance*

Description

A dataset containing the daily realized variance, and some of its lags, obtained from 1-minute close prices of the SP 500. Similar data has been used in the HAR model in Corsi (2009), the HARQ and SHARQ models in Bollerslev et al (2016) and the TVHARQ and TVSHARQ models in Casas et al (2018). The time period runs from Feb 1990 until Dec 2006.

Format

A data frame with 4264 rows and 6 variables.

Date Daily data from Feb 1, 1990 until Dec 29, 2006 - without weekends and days off

RV Daily realized variance at time t

RV_lag Daily realized variance at time t-1

RV_week Weekly average realized variance at time t-5

RV_month Monthly average realized variance at time t-22

RQ_lag_sqrt Daily squared root of the realized quarticity at time t-1

References

Bollerslev, T., Patton, A. J. and Quaedvlieg, R. (2016) Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics*, 192, 1-18.

Bollerslev, T., Tauchen, G. and Zhou, H. (2009) Expected stock returns and variance risk premia. *The Review of Financial Studies*, 22, 44-63.

Casas, I., Mao, X. and Vega, H. (2018) Reexamining financial and economic predictability with new estimators of realized variance and variance risk premium. Url= http://pure.au.dk/portal/files/123066669/rp18_10.pdf

Corsi, F. (2009) A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7, 174-196.

`summary.tvlm` *Print results of functions in tvReg*

Description

Print some results for objects with class attribute `tvlm`, `tvar`, `tvvar`, `tvirf`, `tvsure` and `tvplm`.

Usage

```
## S3 method for class 'tvlm'
summary(object, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvar'
summary(object, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvplm'
summary(object, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvsure'
summary(object, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvvar'
summary(object, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'tvirf'
summary(object, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object`	An object used to select a method.
`digits`	Integer, indicating the minimal number of significant digits.
`...`	Other parameters passed to specific methods.

Details

These functions print a few results from the time-varying estimated coefficients

See Also

`plot.tvlm`, `plot.tvvar`, `plot.tvvar`, `plot.tvirf`,`plot.tvsure`

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

Description

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

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

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

Usage

```
tvAcoef(x)
```

Arguments

x	An object of class `tvvar` generated by `tvVAR`.

Value

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

Examples

```
data(Canada, package="vars")
var.2p <- vars::VAR(Canada, p = 2, type = "const")
tvvar.2p <- tvVAR(Canada, p = 2, type = "const")
A <- vars::Acoef(var.2p)
tvA <- tvAcoef(tvvar.2p)
```

tvAR *Time-Varying Autoregressive Model*

Description

`tvAR` is used to fit an autorregressive model with time varying coefficients.

Usage

```
tvAR(
  y,
  p = 1,
  z = NULL,
  ez = NULL,
  bw = NULL,
  cv.block = 0,
  type = c("const", "none"),
  exogen = NULL,
  fixed = NULL,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  singular.ok = TRUE
)
```

Arguments

`y`	A vector with the dependent variable.
`p`	A scalar indicating the number of lags in the model.
`z`	A vector with the smoothing variable.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`bw`	An opcional 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.
`cv.block`	A positive scalar with the size of the block in leave one block out cross-validation. By default 'cv.block=0' meaning leave one out 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)
`fixed`	(optional) numeric vector of the same length as the total number of parameters. The order of the parameters is intercept (if type = "const"), lags in ascending order and exogenous variables. If supplied, only NA entries in fixed will be estimated.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`singular.ok`	Logical. If FALSE, a singular model is an error.

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 `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` with the following components:

`coefficients`	A vector of dimension obs (obs = number of observations - number lags), with the time-varying coefficients estimates.
`fitted`	The fitted values.
`residuals`	Estimation residuals.
`x`	A matrix of model data, with lagged `y` and exogenous variables.
`y`	A vector with the dependent data used in the model.
`z`	A vector with the smoothing variable in the model.
`ez`	A vector with the smoothing estimation values.
`y.orig`	A vector with the original variable y.
`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
`obs`	Number of observations in estimation.
`totobs`	Number of observations in the original set.
`level`	Confidence interval range.
`runs`	Number of bootstrap replications.
`tboot`	Type of bootstrap.
`BOOT`	List with all bootstrap replications of `coefficients`, if done.

References

Cai, Z. (2007) Trending time-varying coefficient time series with serially correlated errors, *Journal of Econometrics*, 136, pp. 163-188.

Casas, I., Mao, X. and Veiga, H. (2018) Reexamining financial and economic predictability with new estimators of realized variance and variance risk premium. Url= http://pure.au.dk/portal/files/123066669/rp18_10.pdf

Chen, X. B., Gao, J., Li, D., and Silvapulle, P (2017) Nonparametric Estimation and Forecasting for Time-Varying Coefficient Realized Volatility Models. *Journal of Business and Economic Statistics*, 36, 88-100.

Corsi, F. (2009) A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7, 174-196.

See Also

`bw`, `tvLM`, `confint`, `plot`, `print` and `summary`

Examples

```
## Estimate coefficients of different realized variance models
data("RV")
RV2 <- head(RV, 2000)
RV <- RV2$RV
RV_week <- RV2$RV_week
RV_month <- RV2$RV_month
RQ <- RV2$RQ_lag_sqrt
##Corsi (2009) HAR model
HAR <- arima(RV, order = c(1, 0, 0), xreg = cbind (RV_week, RV_month))
print(HAR)

##Chen et al (2017) TVCHAR model
TVCHAR <- tvAR (RV, p = 1, exogen = cbind (RV_week, RV_month), bw = 20)
print(TVCHAR)

##Casas et al (2018) TVHARQ model
TVHARQ <- tvAR (RV, p = 1, exogen = cbind (RV_week, RV_month),
z=RQ, bw = 0.0062)
print(TVHARQ)
```

`tvBcoef` *Coefficient Array of an Estimated tvVAR*

Description

Returns the system estimated coefficients as an array.

Usage

```
tvBcoef(x)
```

Arguments

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

Details

Given an estimated time varying VAR of the form:

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

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)
```

---

tvCor *Time-varying Correlation Estimation*

Description

Estimation of a time-varying/functional coefficients correlation matrix using the local constant or the local linear kernel smoothing methodologies.

Usage

```
tvCor(
  x,
  z = NULL,
  ez = NULL,
  bw = NULL,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian")
)
```

Arguments

x	A matrix.
z	A vector with the variable over which coefficients are smooth over.
ez	(optional) A scalar or vector with the smoothing values. If values are not included then the vector z is used instead.
bw	(optional) A scalar.
cv.block	A positive scalar with the size of the block in leave-one block-out cross-validation. By default 'cv.block=0' meaning leave-one-out cross-validation.
est	A character, either "lc" or "ll" for local constant or local linear.
tkernel	A character, either "Triweight, "Epa" or "Gaussian" kernel functions.

Value

An array of dimension neq x neq x obs.

See Also

tvCov

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()
Rho.hat <-  tvCor(y, bw = 1.4)

##The first time estimate
print(Rho.hat[,,1])
##The mean over time of all estimates
print(apply(Rho.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
Rho.hat <-  tvCor(y, bw = 3.2)
##The first time estimate
print(Rho.hat[,,1])
```

`tvCov` *Time-varying Variance-Covariance Estimation*

Description

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

Usage

```
tvCov(
  x,
  z = NULL,
  ez = NULL,
  bw = NULL,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian")
)
```

Arguments

x	A matrix.
z	A vector with the variable over which coefficients are smooth over.
ez	(optional) A scalar or vector with the smoothing values. If values are not included then the vector z is used instead.

`bw`	(Optional) A scalar.
`cv.block`	A positive scalar with the size of the block in leave-one block-out cross-validation. By default 'cv.block=0' meaning leave-one-out cross-validation.
`est`	A character, either "lc" or "ll" for local constant or local linear.
`tkernel`	A character, either "Triweight, "Epa" or "Gaussian" kernel functions.

Value

An array of dimension neq x neq x obs.

References

Aslanidis, N. and Casas, I (2013) Nonparametric correlation models for portfolio allocation. *Journal of Banking and Finance*, 37, 2268-2283

See Also

`bwCov`, `tvCor`

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
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])
```

`tvFE` *Time-Varying Fixed Effects Estimation*

Description

`tvFE` estimate time-varying coefficient of fixed effects panel data models using kernel smoothing.

Usage

```
tvFE(x, ...)

## S3 method for class 'matrix'
tvFE(
  x,
  y,
  z = NULL,
  ez = NULL,
  bw,
  neq,
  obs,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  ...
)

## S3 method for class 'tvplm'
tvFE(x, ...)
```

Arguments

`x`	An object used to select a method.
`...`	Other arguments passed to specific methods.
`y`	A vector with dependent variable.
`z`	A vector with the variable over which coefficients are smooth over.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`bw`	A numeric vector.
`neq`	A sclar with the number of equations
`obs`	A scalar with the number of time observations
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.

Value

`tvFE` returns a list containing:

`coefficients`	A vector of length obs, number of observations with the time-varying estimates.
`fitted`	A vector of length obs with the fited values from the estimation.
`residuals`	A vector of length obs with the residuals from the estimation.
`alpha`	A vector of length neq with the fixed effects.

`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.

Usage

```
tvGLS(x, ...)

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

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

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

Arguments

`x`	An object used to select a method.
`...`	Other arguments passed to specific methods.
`y`	A matrix.
`z`	A vector with the smoothing variable.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`bw`	A numeric vector.
`Sigma`	An array.
`R`	A matrix.
`r`	A numeric vector.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.

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 three possible kernels are the Triweight, 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 program 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:

`coefficients`	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.

Examples

```
data(FF5F)
x <- list()
## SMALL/LoBM porfolios time-varying three factor model
x[[1]] <- cbind(rep (1, 314), FF5F[, c("NA.Mkt.RF", "NA.SMB",  "NA.HML", "NA.RMW", "NA.CMA")])
x[[2]] <- cbind(rep (1, 314), FF5F[, c("JP.Mkt.RF", "JP.SMB",  "JP.HML", "JP.RMW", "JP.CMA")])
x[[3]] <- cbind(rep (1, 314), FF5F[, c("AP.Mkt.RF", "AP.SMB",  "AP.HML", "AP.RMW", "AP.CMA")])
```

```
x[[4]] <- cbind(rep (1, 314), FF5F[, c("EU.Mkt.RF", "EU.SMB",  "EU.HML", "EU.RMW", "EU.CMA")])
##Returns
y <- cbind(FF5F$NA.SMALL.LoBM, FF5F$JP.SMALL.LoBM, FF5F$AP.SMALL.LoBM,
FF5F$EU.SMALL.LoBM)
##Excess returns
y <- y - cbind(FF5F$NA.RF, FF5F$JP.RF, FF5F$AP.RF, FF5F$EU.RF)
##I fit the data with one bandwidth for each equation
FF5F.fit <- tvGLS(x = x, y = y, bw = c(1.03, 0.44, 0.69, 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,
  impulse = NULL,
  response = NULL,
  n.ahead = 10,
  ortho = TRUE,
  ortho.cov = c("tv", "const"),
  bw.cov = NULL,
  cumulative = FALSE,
  unit.shock = 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.
`unit.shock`	Logical. If TRUE, the impulse responses are calculated for a one unit shock. If FALSE (default), they are calculated for a one standard deviation shock.
`...`	Other parameters passed to specific methods.

Value

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

`irf`	A list of length the number of impulse variable(s). Each element of the list is an array of dim = c(obs x number of response variables x n.ahead).
`Lower`	A list of length the number of impulse variable(s), containing the lower confidence line, if calculated.
`Upper`	A list of length the number of impulse variable(s), containing 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 impuluse response function.
`ortho.cov`	Character, either 'const' or 'tv' (default). This parameter is used when the orthogonal TVIRF 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.
`unit.shock`	Logical, if TRUE the unit impulse response is calculated.

Acknowledgment

The inclusion of the `unit.shock` parameter was motivated by suggestions from Sebin Nidhiri (University of Houston) for suggesting the unit shock feature.

See Also

`bw`, `tvVAR`, `confint`, `plot`, `print` and `summary`

Examples

```
## Not run:
##Inflation rate, unemployment rate and treasury bill
##interest rate for the US as in Primiceri (2005).
data(usmacro, package = "bvarsv")
TVVAR <- tvVAR(usmacro, p = 4, type = "const")

##Estimate a the tvIRF with time-varying covariance function
TVIRF <- tvIRF(TVVAR)

## Estimate tvIRF with a one unit shock instead of one standard deviation
TVIRF_unit <- tvIRF(TVVAR, unit.shock = TRUE)

##Cumulative impulse response function
TVIRF2 <- tvIRF(TVVAR, cumulative = TRUE)


## End(Not run)
```

`tvLM` *Time-Varying Coefficients Linear Models*

Description

`tvLM` is used to fit a time-varying coefficients linear model

Usage

```
tvLM(
  formula,
  z = NULL,
  ez = NULL,
  data,
  bw = NULL,
  cv.block = 0,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  singular.ok = TRUE
)
```

Arguments

`formula`	An object of class formula.
`z`	A vector with the smoothing variable.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.

`data`	An optional data frame or matrix.
`bw`	An opcional scalar. It represents the bandwidth in the estimation of trend coefficients. If NULL, it is selected by cross validation.
`cv.block`	A positive scalar with the size of the block in leave one block out cross-validation. By default 'cv.block=0' meaning leave one out cross-validation.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`singular.ok`	Logical. If FALSE, a singular model is an error.

Details

Models for `tvLM` are specified symbolically using the same formula format than function `lm`. A typical model has the form *response ~ 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 ~ x - 1 or y ~ 0 + x.

Value

An object of class `tvlm` The object of class `tvlm` have the following components:

`coefficients`	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.
`ez`	A vector with the smoothing estimation 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 `coefficients`, if done.

References

Bollerslev, T., Patton, A. J. and Quaedvlieg, R. (2016) Exploiting the errors: A simple approach for improved volatility forecasting. *Journal of Econometrics*, 192, 1-18.

Casas, I., Mao, X. and Veiga, H. (2018) Reexamining financial and economic predictability with new estimators of realized variance and variance risk premium. Url= http://pure.au.dk/portal/files/123066669/rp18_10.pdf

See Also

bw, tvAR, confint, plot, print and summary

Examples

```
## Simulate a linear process with time-varying coefficient
## as functions of scaled time.
set.seed(42)
tau <- seq(1:200)/200
beta <- data.frame(beta1 = 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
tvlm.fit <- tvLM(y ~ 0 + X1 + X2, data = data, bw = 0.29)

## Estimate coefficients of different realized variance models
data("RV")
RV2 <- head(RV, 2000)
##Bollerslev t al. (2016) HARQ model
HARQ <- with(RV2, lm(RV ~ RV_lag + I(RV_lag * RQ_lag_sqrt) + RV_week + RV_month))

#Casas et al. (2018) TVHARQ model
TVHARQ <- with(RV2, tvLM (RV ~ RV_lag + RV_week + RV_month, z = RQ_lag_sqrt,
                         bw = 0.0061))
boxplot(data.frame(TVHARQ = TVHARQ$coefficients[,2] * RV2$RV_lag,
                   HARQ = (HARQ$coef[2] + HARQ$coef[3] * RV2$RQ_lag_sqrt)*RV2$RV_lag),
                   main = expression (RV[t-1]), outline = FALSE)
```

---

tvOLS *Time-Varying Ordinary Least Squares*

Description

tvOLS estimate time-varying coefficient of univariate linear models using the kernel smoothing OLS.

Usage

```
tvOLS(x, ...)

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

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

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

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

Arguments

`x`	An object used to select a method.
`...`	Other arguments passed to specific methods.
`y`	A vector with dependent variable.
`z`	A vector with the variable over which coefficients are smooth over.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`bw`	A numeric vector.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`singular.ok`	Logical. If FALSE, a singular model is an error.

Value

`tvOLS` returns a list containing:

`coefficients`	A vector of length obs, number of observations time observations.
`fitted`	A vector of length obs with the fitted values from the estimation.
`residuals`	A vector of length obs with the residuals from the estimation.

See Also

`bw` for bandwidth selection, `tvLM` and `tvAR`.

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
coef.lm <- stats::lm(y~0+X1+X2, data = X)$coef
coef.tvlm <-  tvOLS(x = as.matrix(X), y = y, bw = 0.1)$coefficients
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)
```

---

tvPhi *Time-Varying Coefficient Arrays of the MA Represention*

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, ...)
```

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.

---

tvPLM *Time-Varying Coefficients Panel Data Models*

Description

Fits a balanced panel data model using the Time-Varying Pooled Ordinary Least Squares, the Time-Varying Random Effects and the Time-Varying Fixed Effects models.

Usage

```
tvPLM(
  formula,
  z = NULL,
  ez = NULL,
  data,
  index = NULL,
  bw = NULL,
  bw.cov = NULL,
  cv.block = 0,
  method = c("pooling", "random", "within"),
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  control = tvreg.control(...),
  ...
)
```

Arguments

`formula`	An object of class formula.
`z`	A vector containing the smoothing variable.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`data`	An optional data frame or matrix.
`index`	Indicates the individual and time indexes.
`bw`	An opcional scalar. It represents the bandwidth in the estimation of trend coefficients. If NULL, it is selected by cross validation.
`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 for method `"random"`.
`cv.block`	A positive scalar with the size of the block in leave one block out cross-validation. By default 'cv.block=0' meaning leave one out cross-validation.
`method`	A character with the choice of panel model/estimation method: If `method = "pooling"` (default) then the data is pooled estimated with time-varying OLS. No individual or time effects are estimated If `method = "random"` then individual effects are considered random and independent of the regressors. If `method = "within"` then individual effects which might be correlated with the regressors are estimated.
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`control`	list of control parameters. The default is constructed by the function `tvreg.control`. See the documentation of `tvreg.control` for details.
`...`	Other parameters passed to specific methods.

Details

This function wraps up the kernel smoothing time-varying coefficient pooled, random effects and fixed effects estimators.

Bandwidth selection is of great importance in kernel smoothing methodologies and it is done automatically by cross-validation.

A panel data model consists of "neq" elements in the cross-sectional dimention and "obs" number of time observations for each cross-section. All variables are the same for each equation which have common coefficients.

Value

`tvPLM` returns a list of the class `tvplm` containing the results of model, results of the estimation and confidence instervals if chosen. The object of class `tvplm` have the following components:

`coefficients`	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.
`Lower`	If `level` non equal zero, an array of dimension obs x nvar x neq containing the confidence interval lower band.
`Upper`	If `level` non equal zero, an array of dimension obs x nvar x neq containing the confidence interval upper band.
`fitted`	The fitted values.
`residuals`	Estimation residuals.
`x`	A list with the regressors data.
`y`	A matrix with the dependent variable data.
`z`	A vector with the smoothing variable.
`ez`	A vector with the smoothing estimation values.
`alpha`	A vector with the individual fixed effects, if chosen.
`bw`	Bandwidth of mean estimation.
`totobs`	Integer specifying the total number of observations.
`neq`	Integer specifying the number of cross-section observations.
`obs`	Integer specifying the number of time observations per cross-section.
`nvar`	Number of variables.
`method`	Estimation method.
`est`	Nonparemtric estimation methodology.
`tkernel`	Kernel type.
`level`	Confidence interval range.
`runs`	Number of bootstrap replications.
`tboot`	Type of bootstrap.
`BOOT`	List with all bootstrap replications of `coefficients`, if done.
`formula`	Initial formula.
`call`	Matched call.

References

Casas, I., Gao, J., Peng, B. and Xie, S. (2021). Time-Varying Income Elasticities of Healthcare Expenditure for the OECD and Eurozone. *Journal of Applied Econometrics, 36, pp. 328-345*.

Sun, Y., Carrol, R.J and Li, D. (2009). Semiparametric Estimation of Fixed-Effects Panel Data Varying Coefficient Models. *Advances in Econometrics*, 25, pp. 101-129.

See Also

bw, confint, plot, print and summary

Examples

```
data(OECD)
##TVPOLS estimation of the model
tvpols <- tvPLM(lhe~lgdp+pop65+pop14+public, index = c("country", "year"),
 data = OECD, method ="pooling", bw = 0.3)
## Not run:
tvfe <- tvPLM(lhe~lgdp+pop65+pop14+public, index = c("country", "year"),
 data = OECD, method ="within", bw = 0.8)
tvre <- tvPLM(lhe~lgdp+pop65+pop14+public, index = c("country", "year"),
 data = OECD, method ="random", bw = 0.3)

## End(Not run)
```

---

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

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

Arguments

`x`	An object of class `tvvar`, generated by tvVAR().
`nstep`	An integer specifying the number of othogonalised moving error coefficient matrices to be calculated for each time t.
`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.
`unit.shock`	Logical. If TRUE, the impulse responses are calculated for a one unit shock. If FALSE (default), they are calculated for a one standard deviation shock.
`...`	Other parameters passed to specific methods.

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).

`tvRE` *Time-Varying Random Effects Estimation*

Description

`tvRE` estimate time-varying coefficient of a random effects panel data model using kernel smoothing.

Usage

```
tvRE(x, ...)

## S3 method for class 'matrix'
tvRE(
  x,
  y,
  z = NULL,
  ez = NULL,
  bw,
  Sigma = NULL,
  neq,
  obs,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  ...
)

## S3 method for class 'tvplm'
tvRE(x, ...)
```

Arguments

`x`	An object used to select a method.
`...`	Other arguments passed to specific methods.
`y`	A vector with dependent variable.
`z`	A vector with the variable over which coefficients are smooth over.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`bw`	A numeric vector with the bandwidth.
`Sigma`	NULL (default) or a matrix of size obs x obs..
`neq`	A scalar with the number of equations
`obs`	A scalar with the number of time observations
`est`	The nonparametric estimation method, one of "lc" (default) for linear constant or "ll" for local linear.
`tkernel`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.

Value

`tvRE` returns a list containing:

`coefficients`	A vector of length obs, number of observations with the time-varying estimates.
`fitted`	A vector of length obs with the fited values from the estimation.
`residuals`	A vector of length obs with the residuals from the estimation.
`alpha`	A vector of length neq with the fixed effects.

`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,
  ez = NULL,
  bw = NULL,
  cv.block = 0,
  data,
  method = c("tvOLS", "tvFGLS", "tvGLS"),
  Sigma = NULL,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  bw.cov = NULL,
  singular.ok = TRUE,
  R = NULL,
  r = NULL,
  control = tvreg.control(...),
  ...
)
```

Arguments

`formula`	A list of formulas, one for each equation.
`z`	A vector containing the smoothing variable.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`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.
`cv.block`	A positive scalar with the size of the block in leave one block out cross-validation. By default 'cv.block = 0' meaning leave one out 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`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`bw.cov`	An optional scalar. It represents the bandwidth in the nonparametric estimation of the 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 `tvreg.control`. See the documentation of `tvreg.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 `bw` and `tvGLS` separatedly.

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 $\beta_t = \left(\beta'_{1t}, \ldots, \beta'_{neqt}\right)'$ 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 $E(u_t u'_t) = \Sigma_t$.

Value

`tvSURE` returns a list of the class `tvsure` containing the results of the whole system, results of the estimation and confidence instervals if chosen. The object of class `tvsure` have the following components:

`coefficients`	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.
`Lower`	If `level` non equal zero, an array of dimension obs x nvar x neq containing the confidence interval lower band.
`Upper`	If `level` non equal zero, an array of dimension obs x nvar x neq containing the confidence interval upper band.
`Sigma`	An array of dimension obs x neq x neq with the estimates of the errors covariance matrix.
`fitted`	The fitted values.
`residuals`	Estimation residuals.
`x`	A list with the regressors data.
`y`	A matrix with the dependent variable data.
`z`	A vector with the smoothing variable.
`ez`	A vector with the smoothing estimation values.
`bw`	Bandwidth of mean estimation.
`obs`	Integer specifying the number of observations in each equation (balanced sample).

`neq`	Integer specifying the number of equations.
`nvar`	Vector of integers specifying the number of variables in each equation.
`method`	Estimation method.
`est`	Nonparemtric estimation methodology.
`tkernel`	Kernel type.
`bw.cov`	Bandwidht of Sigma estimation.
`level`	Confidence interval range.
`runs`	Number of bootstrap replications.
`tboot`	Type of bootstrap.
`BOOT`	List with all bootstrap replications of `coefficients`, if done.
`R`	Restrictions matrix.
`r`	Restrictions vector.
`formula`	Initial formula.

References

Casas, I., Ferreira, E., and Orbe, S. (2021). Time-Varying Coefficient Estimation in SURE Models: Application to Portfolio Management. *Journal of Financial Econometrics*, 19, 707-745.

Chen, X. B., Gao, J., Li, D., and Silvapulle, P (2017) Nonparametric Estimation and Forecasting for Time-Varying Coefficient Realized Volatility Models. *Journal of Business and Economic Statistics*, pp.1-13

Granger, C. W (2008) Non-Linear Models: Where Do We Go Next - Time Varying Parameter Models? *Studies in Nonlinear Dynamics and Econometrics*, 12, pp. 1-11.

Kristensen, D (2012) Non-parametric detection and estimation of structural change. *Econometrics Journal*, 15, pp. 420-461.

Orbe, S., Ferreira, E., and Rodriguez-Poo, J (2004) On the estimation and testing of time varying constraints in econometric models, *Statistica Sinica*.

See Also

`bw`, `tvCov`, `tvVAR`, `confint`, `plot`, `print` and `summary`

Examples

```
## Not run:
data("Kmenta", package = "systemfit")
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list(demand = eqDemand, supply = eqSupply)
eqSupply2 <- consump ~  price + farmPrice
system2 <- list(demand = eqDemand, supply = eqSupply2)

##OLS estimation of a system
OLS <- systemfit::systemfit(system, method = "OLS", data = Kmenta)
##tvOLS estimation of a system with the local linear estimator
```

```
##removing trend because it is included in the intercept changing over time
TVOLS <- tvSURE(system2, data = Kmenta,  est = "ll")

##SUR/FGLS estimation
FGLS <- systemfit::systemfit(system, data = Kmenta, method = "SUR")
##tvSURE estimation
TVFGLS <- tvSURE(system, data = Kmenta, method = "tvFGLS")

## End(Not run)
```

tvVAR — *Time-varying Vector Autoregressive Models*

Description

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

Usage

```
tvVAR(
  y,
  p = 1,
  z = NULL,
  ez = NULL,
  bw = NULL,
  cv.block = 0,
  type = c("const", "none"),
  exogen = NULL,
  est = c("lc", "ll"),
  tkernel = c("Triweight", "Epa", "Gaussian"),
  singular.ok = TRUE
)
```

Arguments

`y`	A matrix with dimention 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.
`ez`	(optional) A scalar or vector with the smoothing values. If values are not included then the vector `z` is used instead.
`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.
`cv.block`	A positive scalar with the size of the block in leave one block out cross-validation. By default 'cv.block = 0' meaning leave one out 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`	A character, either "Triweight" (default), "Epa" or "Gaussian" kernel function.
`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:

`coefficients`	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.
`z`	A vector with the smoothing variable.
`ez`	A vector with the smoothing estimation values.
`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.

References

Casas, I., Ferreira, E., and Orbe, S. (2021). Time-Varying Coefficient Estimation in SURE Models: Application to Portfolio Management. *Journal of Financial Econometrics*, 19, 707-745.

Primiceri, G.E. (2005) Time varying structural vector autoregressions and monetary policy. *Review of Economic Studies*, 72, 821-852.

See Also

`bw`, `tvIRF`, `plot`, `print` and `summary`

Examples

```
##Inflation rate, unemployment rate and treasury bill interest rate for
##the US, as used in Primiceri (2005).
data(usmacro, package = "bvarsv")
VAR.fit <- vars::VAR(usmacro, p = 6, type = "const")
tvVAR.fit <- tvVAR(usmacro, p = 6, type = "const", bw = c(1.8, 20, 20))
plot(tvVAR.fit)
```

`update.tvlm` *Update and Re-fit the Models of package tvReg*

Description

Update and Re-fit the Models of package tvReg

Usage

```
## S3 method for class 'tvlm'
update(object, ...)

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

## S3 method for class 'tvvar'
update(object, ...)

## S3 method for class 'tvsure'
update(object, ...)

## S3 method for class 'tvplm'
update(object, ...)
```

Arguments

`object`	An object of any class in package tvReg.
`...`	Other parameters passed to specific methods.

Value

An object of the same class than the argument *object*.

Index

∗ **datasets**
    CEES, 6
    FF5F, 10
    OECD, 13
    RV, 18

bw, 3, *22*, *31*, *34*, *35*, *39*, *43*, *44*, *46*
bwCov, 5, *26*

CEES, 6
confint, *2*, *22*, *31*, *34*, *39*, *44*
confint.tvar (confint.tvlm), 7
confint.tvirf (confint.tvlm), 7
confint.tvlm, 7
confint.tvplm (confint.tvlm), 7
confint.tvsure (confint.tvlm), 7
confint.tvvar (confint.tvlm), 7

FF5F, 10
fitted, *2*
forecast, *2*, 11, *16*

OECD, 13

plot, *2*, *22*, *31*, *34*, *39*, *44*, *46*
plot.tvar, *18*
plot.tvar (plot.tvsure), 14
plot.tvirf, *18*, *19*
plot.tvirf (plot.tvsure), 14
plot.tvlm, *18*, *19*
plot.tvlm (plot.tvsure), 14
plot.tvplm, *18*
plot.tvplm (plot.tvsure), 14
plot.tvsure, 14, *18*, *19*
plot.tvvar, *18*, *19*
plot.tvvar (plot.tvsure), 14
predict, *2*, *12*
predict.tvar (predict.tvlm), 15
predict.tvlm, 15
predict.tvplm (predict.tvlm), 15
predict.tvsure (predict.tvlm), 15
predict.tvvar (predict.tvlm), 15
print, *2*, *22*, *31*, *34*, *39*, *44*, *46*
print.tvar (print.tvlm), 17
print.tvirf (print.tvlm), 17
print.tvlm, 17
print.tvplm (print.tvlm), 17
print.tvsure (print.tvlm), 17
print.tvvar (print.tvlm), 17

resid, *2*
RV, 18

summary, *2*, *22*, *31*, *34*, *39*, *44*, *46*
summary (summary.tvlm), 19
summary.tvlm, 19

tvAcoef, 20
tvAR, *2*, *9*, *15*, 20, *34*, *35*
tvar (tvAR), 20
tvar-class (tvAR), 20
tvBcoef, 23
tvCor, 24, *26*
tvCov, *24*, 25, *44*
tvFE, *2*, 26
tvGLS, *2*, 28, *43*
tvIRF, *2*, 30, *46*
tvirf-class (tvIRF), 30
tvirf. (tvIRF), 30
tvLM, *2*, *9*, *15*, *22*, 32, *35*
tvlm (tvLM), 32
tvlm-class (tvLM), 32
tvOLS, *2*, 34
tvPhi, 36
tvPLM, *2*, *15*, 36
tvplm (tvPLM), 36
tvplm-class (tvPLM), 36
tvPsi, 39
tvRE, *2*, 40
tvReg (tvReg-package), 2
tvReg-package, 2

tvreg.control, *37*, *42*
tvSURE, *2*, *9*, *15*, 41
tvsure *(*tvSURE*)*, 41
tvsure-class *(*tvSURE*)*, 41
tvVAR, *2*, *9*, *15*, *20*, *23*, *31*, *44*, 45
tvvar *(*tvVAR*)*, 45
tvvar-class *(*tvVAR*)*, 45

update.tvar *(*update.tvlm*)*, 47
update.tvlm, 47
update.tvplm *(*update.tvlm*)*, 47
update.tvsure *(*update.tvlm*)*, 47
update.tvvar *(*update.tvlm*)*, 47