tvReg: Time-Varying Coefficients Linear Regression for Single and Multiple Equations

Isabel Casas & Ruben Fernandez-Casal

2017-11-03

Abstract

The source code of the *tvReg* package is publicly available for download from the Comprehensive *R* Archive Network (CRAN, https://CRAN.R-project.org/). The five basic functions in this package are *tvLM*, *tvAR*, *tvSURE*, *tvVAR* and *tvIRF*. Moreover, this package provides tools for graphical display of the results, extract residuals and fitted values, calculate bandwidths, time-varying variance-covariance matrix of the error term and two estimators: the time-varying ordinary least squares programmed in function *tvOLS* and the time-varying generalised least squares in function *tvGLS*.

Contents

Univariate models: the tv-LM and tv-AR

The general formula of these models is:

$$y_t = X_t'\beta_t + u_t \; t = 1, \dots, T \quad (1)$$

where $X_t = (X_{0t}, X_{1t}, \dots, X_{dt})'$ and $\beta_t = (\beta_{0t}, \beta_{1t}, \dots, \beta_{dt})'$ are vectors and u_t is a random variable. When X_t contains lags of y_t then we have the time-varying parameters AR model (tv-AR) which is fitted by function *tvAR*. When X_t contains only exogenous variables, then the model is denoted by tv-LM and it is fitted by function *tvLM*.

Standard usage of *tvLM*

The *tvLM* function fits a tv-LM using function *tvOLS* in the estimation. The latter includes the algorithm of kernel smoothing nonparametric techniques for a linear model. The only mandatory argument is *formula* which should be a single formula for a single equation model. This formula follows the standard regression formula in *R* (see documentation of *formula*). The returned object of this function is an object of class *tvlm*.

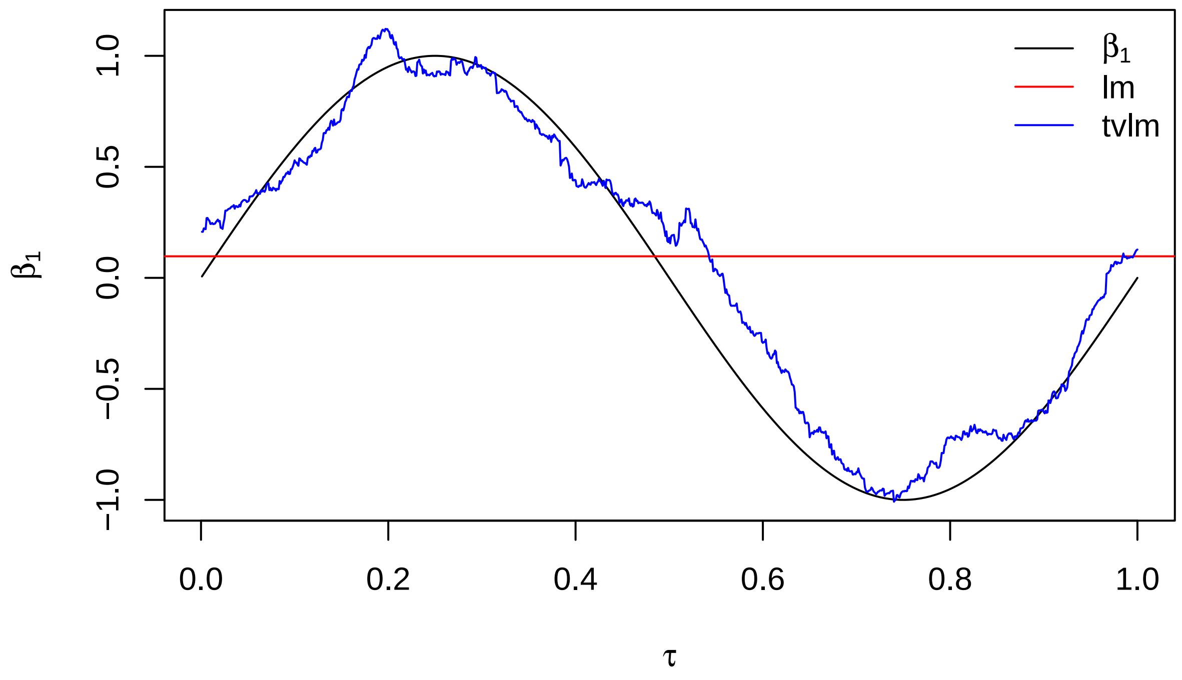
The following example demonstrates the functionality of *tvLM* on generated data and compares it its *lm* estimation. The latter assumes that the model coefficients are constant. The data generating process of the example below has time-varying coefficients and therefore its estimation with function *lm* will only find some kind of middle value of all the values over time.

We want to estimate the following model:

$$y_t = \beta_{1t} X_{1t} + \beta_{2t} X_{2t} + u_t, \quad t = 1, \ldots, T,$$

where the coefficients are function of a scaled time variable $\tau = t/T$. In particular, $\beta_{1t} = \sin(2\pi\tau)$ and $\beta_{2t} = 2\tau$.

```
## Simulate a linear process with time-varying coefficient as functions of
## scaled time.
library(tvReg)
tau <- seq(1:1000)/1000
beta <- data.frame(beta1 = sin(2 * pi * tau), beta2 = 2 * tau)
X1 <- rnorm(1000)
X2 <- rchisq(1000, df = 4)
error <- rt(1000, 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)

## Plot the estimates of beta1
plot(tau, beta[, 1], type = "l", main = "", ylab = expression(beta[1]), xlab = expression(tau),
    ylim = range(beta[, 1], model.tvlm$tvcoef[, 1]))
abline(h = coef.lm[1], col = 2)
lines(tau, model.tvlm$tvcoef[, 1], col = 4)
legend("topright", c(expression(beta[1]), "lm", "tvlm"), col = c(1, 2, 4), bty = "n",
    lty = 1)
```



The first line loads the *tvReg* package. Then data is simulated and a data frame is created with the dependent variable and the regressors. Estimation of this model is run with the *lm* and the *tvLM* functions for comparison. As we see in the plot, the estimation assuming a constant β_{1t} lays around the average value of all β_{1t}, while the estimation assuming time-varying coefficients is more accurate and closer to the true value.

User options of *tvLM*

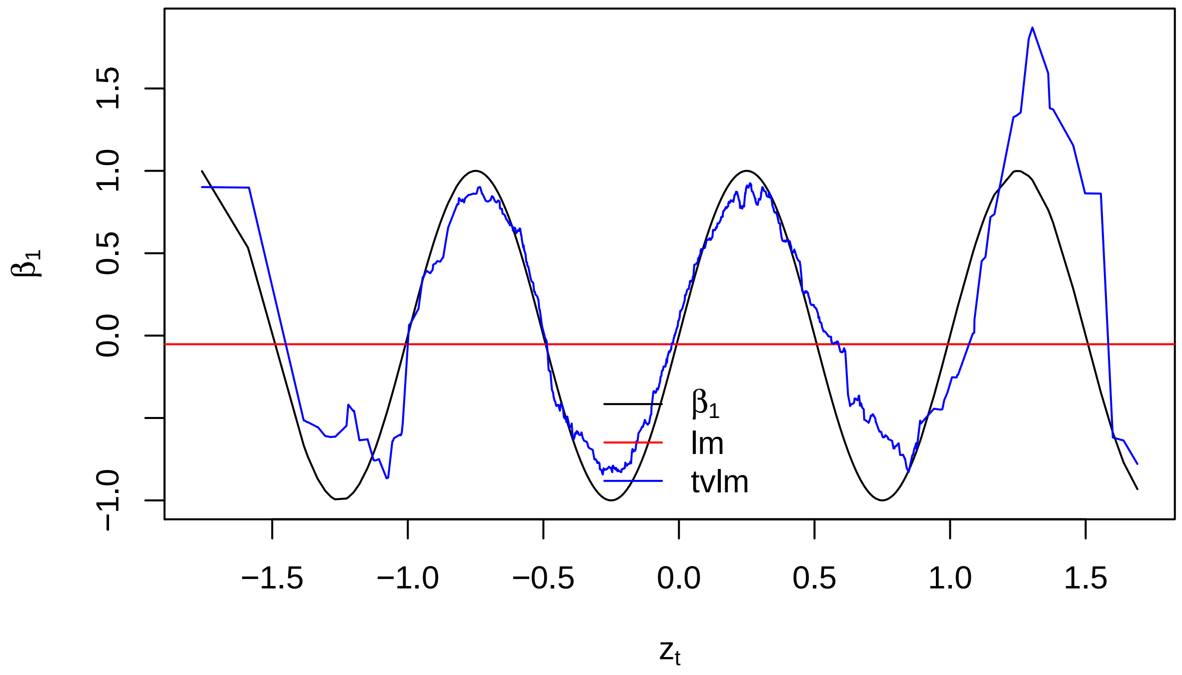
The user can provide additional optional arguments to modify the default estimation. All optional arguments are described below:

Smoothing random variable

By default, coefficients are assumed to vary as an unknown function of the scaled time (τ in the example). There are contexts in which coefficients depend on another random variable. The user can modify this setting by entering a numeric vector in parameter z with the values of this smoothing variable over the correspondent time period. The example below shows this functionality with generated data. The example generating process of function *arima.sim* from package *stats* is used in the example.

```
## Data generation
set.seed(42)
z <- stats::arima.sim(n = 1000, list(ar = c(0.8897, -0.4858), ma = c(-0.2279,
    0.2488)), sd = sqrt(0.1796))
beta <- data.frame(beta1 = sin(2 * pi * z), beta2 = 2 * z)
y <- apply(cbind(X1, X2) * beta, 1, sum) + error
data <- data.frame(y = y, X1 = X1, X2 = X2, z = z)
```

```r
## Coefficients estimation
coef.lm <- stats::lm(y ~ 0 + X1 + X2, data = data)$coef
model.tvlm2 <- tvLM(y ~ 0 + X1 + X2, z = z, data = data, bw = 0.4, est = "ll")

## Plotting the estimates of beta1
sort.index <- sort.int(z, index.return = TRUE)$ix
plot(z[sort.index], beta[sort.index, 1], type = "l", main = "", ylab = expression(beta[1]),
    xlab = expression(z[t]), ylim = range(beta[, 1], model.tvlm2$tvcoef[, 1]))
abline(h = coef.lm[1], col = 2)
lines(z[sort.index], model.tvlm2$tvcoef[sort.index, 1], col = 4)
legend("bottom", c(expression(beta[1]), "lm", "tvlm"), col = c(1, 2, 4), bty = "n",
    lty = 1)
```



As in the previous example, the coefficient estimate from *lm* falls around the mean of all β_{1t} over the whole time period. While the *tvLM* estimates follow the true process closely. Note that if the bandwidth was very large, then the two estimates would be very similar.

Bandwidth

When no bandwidth value is input in parameter *bw*, this is calculated by cross-validation. This minimisation might be a bit slow when the dataset is large and it should be avoided when the user knows the value of the bandwidth. As we can see in the previous example, the bandwidth was chosen to be 0.4 by the user.

Kernel type

The default kernel is the Epanechnikov kernel ("Epa"), a compact kernel function in [-1, 1]. The Gaussian kernel may also be chosen in *tkernel* as well, although the authors do not recommend it as calculations are computationally slower.

Estimation methodology

The default estimation methodology is the Nadaraya-Watson or local constant ("lc"). Parameter *est* can be chosen to be "ll" to perform a local linear estimation.

Data

Function *tvLM* has also argument *data* to specify the data frame or matrix that contains the variables in the model. If no dataset is input, the function uses variables in the global environment or the one attached to the search path.

Singular fit

As the *tvOLS* wraps R function *lm.wfit*, the default is not to refuse to fit a low-rank model and have NA elements in the coefficients. The user might change parameter *singular.ok* to *FALSE*, so the programme stops in case of low-rank model.

Standard usage of *tvAR*

A tv-AR model is a special case of tv-LM in which regressors contain lagged variables of the dependent variable *y*. The number of lags included depend on the model order *p*. Other exogenous variables might be included in the model using parameter *exogen*. An intercept is included by default unless the user inputs *type* = "none" into the function call. Econometrically, this function also wraps estimator *tvOLS* that needs a bandwidth *bw* which is calculated by cross-validation when the user does not input any number. The return is an object of class *tvar*.

The following examples demonstrates the functionality of *tvAR* on generated data, and compares it the output of function *ar.ols* from the *stats* package. The data generating process of the example below has time-varying coefficients and therefore its estimation with function *ar.ols* will only find a middle value of all the values over time.

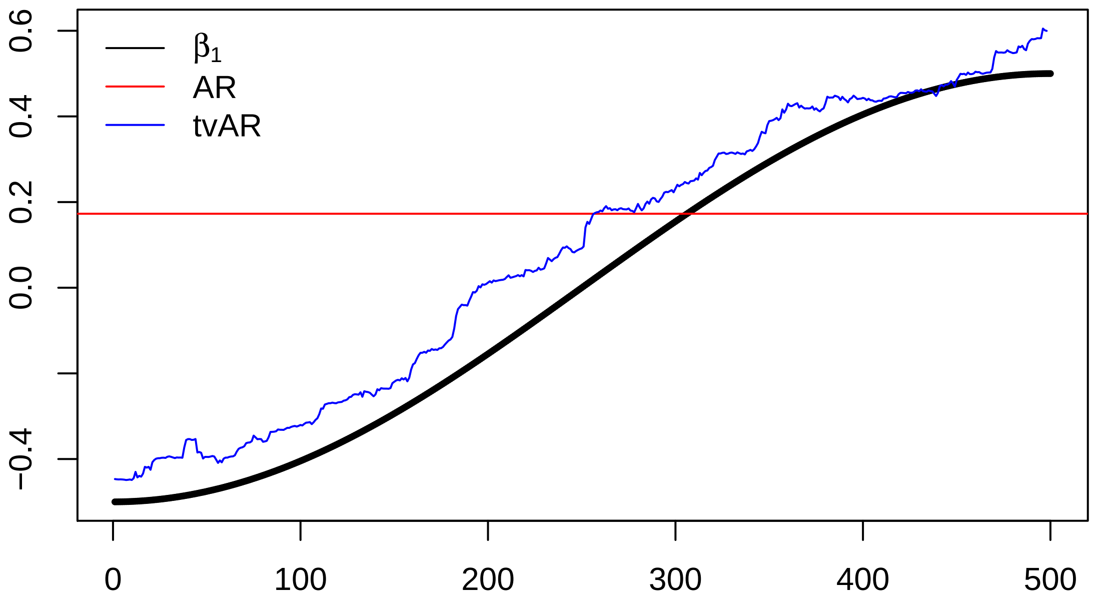
The data generating process of the example below is as following:

$$y_t = \beta_{1t} y_{t-1} + \beta_{2t} y_{t-2} + u_t, \quad t = 1, \ldots, T,$$

with coefficients $\beta_{1t} = 0.5\cos(2\pi\tau)$ and $\beta_{2t} = (\tau - 0.5)^2$ where $\tau = t/T$.

```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)
}
```

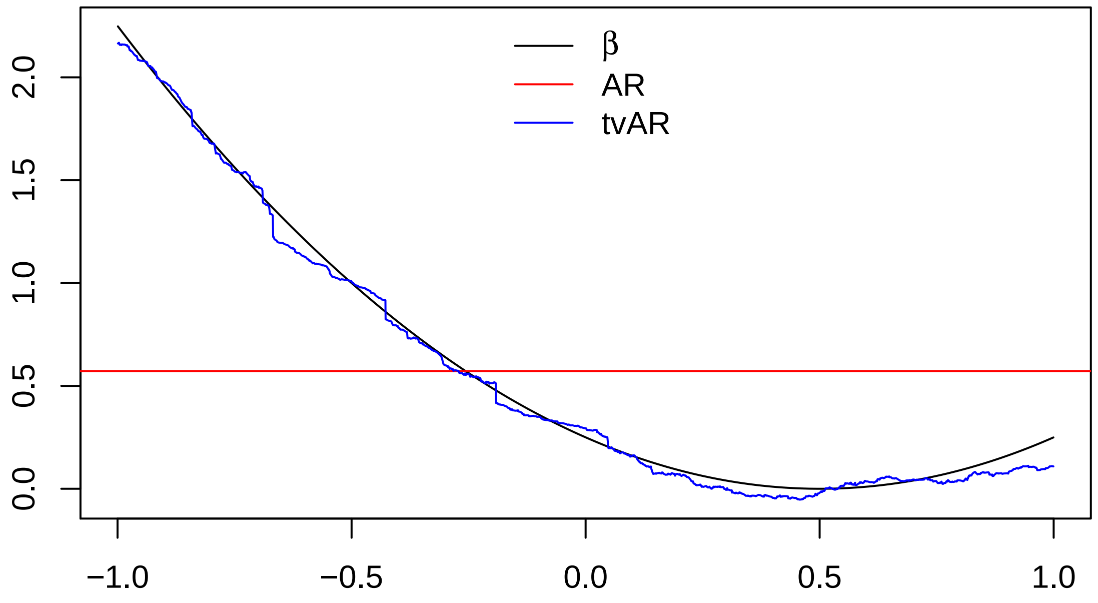
```
Y <- tail(y, 500)

## Estimate coefficients of process Y with ar.ols and tvAR and compare them
## in a plot
model.ar.2p <- ar.ols(Y, aic = FALSE, order = 2, intercept = FALSE, demean = FALSE)
model.tvAR.2p <- tvAR(Y, p = 2, type = "none", est = "ll")
plot(tail(beta[, 1], 500), ylim = range(model.tvAR.2p$tvcoef[, 1], tail(beta[,
    1], 500)), xlab = "", ylab = "", cex = 0.5, pch = 20)
abline(h = model.ar.2p$ar[1], col = 2)
lines(model.tvAR.2p$tvcoef[, 1], col = 4)
legend("topleft", c(expression(beta[1]), "AR", "tvAR"), col = c(1, 2, 4), lty = 1,
    bty = "n")
```



The coefficients might change in time as a function of a random variable z_t. See example below for illustration:

```
## Simulate a AR(1) process with coefficients depending on z
z <- runif(2000, -1, 1)
beta <- (z - 0.5)^2
y <- numeric(2000)
y[1] <- 0.5
error <- rnorm(2000)
## y(t) = beta1(z(t)) y(t-1) + ut
for (t in 2:2000) {
    y[t] <- y[(t - 1)] %*% beta[t] + error[t]
}
## Remove initial conditions effects
```

```
Z <- z  #tail (z, 1500)
Y <- y  #tail (y, 1500)
# beta <- tail (beta, 1500) Estimate coefficients of process Y with ar.ols
# and tvAR and compare them in a plot
model.ar.1p <- ar.ols(Y, aic = FALSE, order = 1, intercept = FALSE, demean = FALSE)
model.tvAR.1p.z <- tvAR(Y, p = 1, z = Z, type = "none", est = "ll")

## Plot comparison of AR and tvAR estimates. Random variable Z must be
## ordered for a meaningful plot
index <- sort.int(model.tvAR.1p.z$z, index.return = TRUE)$ix
z.sort <- model.tvAR.1p.z$z[index]
beta <- tail(beta, length(z.sort))[index]
beta.hat <- model.tvAR.1p.z$tvcoef[index, ]
plot(z.sort, beta, ylim = range(beta.hat, beta), xlab = "", ylab = "", type = "l")
abline(h = model.ar.1p$ar[1], col = 2)
lines(z.sort, beta.hat, col = 4)
legend("top", c(expression(beta), "AR", "tvAR"), col = c(1, 2, 4), lty = 1,
    bty = "n")
```



User options of *tvAR*

The user can provide additional optional arguments to modify the default estimation. Refer to section 'User options of *tvLM*' to understand function parameters *bw*, *est*, *tkernel* and *singular.ok* for details. Also, the smoothing variable *z* maybe a vector of the scale of time or a random variable. In addition, the *tvAR* has the following arguments:

Type

The fitted model contains an intercepted by default, i.e., it has a mean different from zero. The user can set parameter *type* to ‘none’ so the model has a zero mean.

Autoregressive model with coefficient restrictions

An autoregressive process of order p does not necessarily contains all lags coefficients, some lags might not have an effect in the current state. Parameter *fixed* permits to impose these restrictions. The variables order in this function is: lags variables, exogenous variables (if any) and intercept (if any). For example in a model of lag 4, the order is: lag 1, lag 2, lag 3, lag 4, exogenous variables and mean. Parameter *fixed* is a vector as long as the number of parameters in our model and it should contain an NA for the parameters we want to estimate and zero for the constrained lags.

The example below uses simulated variable Y from above and fits a model with 6 lags, from which only the first, third and sixth lag are part of the model, the model has an intercept and there are no exogenous variables. It is estimated with the nonparametric local linear model and the bandwidth is calculated by cross-validation.

```
## Estimate only coefficient from odd lags and the intercept
tvAR.6p <- tvAR(y, p = 6, type = "const", fixed = c(NA, 0, NA, 0, NA, 0, NA),
    est = "ll")
```

Systems of linear models: the tv-VAR and tv-SURE models

The formula of systems of linear models can be generally express as in (1) with $y_t = (y_{1t}, y_{2t}, \ldots, y_{Kt})'$ for K the number of equations in the system. Similarly, the error term $u_t = (u_{1t}, u_{2t}, \ldots, u_{Kt})'$. Variables X_t and β_t are matrices.

Regarding systems of equations, the *tvSURE* function fits a time-varying seemingly unrelated equations models. Its main parameter is *formula* which is a list of objects of class *formula*, one formula for each equation. Equations might have a different number of regressors and these do not have to be the same.

Another multiequation model included in the package is the time-varying vector autoregressive in function *tvVAR* whose main parameters are a matrix of dependent variables y and the number of lags p. It returns an object of type *tvvar*. Coefficients of the *tvVAR* are not interpretable, instead the time-varying coefficients impulse response function is coded in function *tvIRF*. The latter takes an object of class *tvvar* and returns an object of type *tvirf*.

Standard usage of *tvVAR* and *tvIRF*

A time-varying coefficients vector autoregressive model is a multiequation time-varying coefficients autoregressive model. The dependent variable y is a matrix with as many columns as equations. This model has at least two equations. Regressors are often the same for each equation and they consist of the lagged variables of y, where the number of lags is given in parameter p, an intercept if parameter *type* is set to “const” and other exogenous variables input in parameter *exogen*. Econometrically, function *tvOLS* is called to calculate the estimates for each equation independently using *bw* as the bandwidth. The user can input a single scalar in *bw* or a vector of scalars, one bandwidth for each equation. If *bw* is kept as *NULL*, one bandwidth for each equation is calculated by cross-validation. The return is an object of class *tvsure*.

As in the other models, the kernel and the estimation methodology can be chosen in parameters *tkernel* and *est*. The return is an object of class *tvvar* which can be used to estimate the time-varying impulse response function with function *tvIRF*. The smoothing variable by default is time, but another random variable can be used by inputting it in parameter *z*.

Example below uses the macroeconomic data from Primiceri (2005). Three variables are included in dataset *usmacro* from package *bvarsv*: inflation rate (inf), unemployment rate (une) and treasury bill interest rate (tbi) for the US. A constant parameters VAR is estimated using function *VAR* from package *vars* and a time-varying coefficients VAR is estimated with *tvVAR* both with four lags.

$$\text{inf}_t = a^1_t + \sum_{i=1}^{4} b^1_{it}\ \text{inf}_{t-i} + \sum_{i=1}^{4} c^1_{it}\ \text{une}_{t-i} + \sum_{i=1}^{4} d^1_{it}\ \text{tbi}_{t-i} + u^1_t$$

$$\text{une}_t = a^2_t + \sum_{i=1}^{4} b^2_{it}\ \text{inf}_{t-i} + \sum_{i=1}^{4} c^2_{it}\ \text{une}_{t-i} + \sum_{i=1}^{4} d^2_{it}\ \text{tbi}_{t-i} + u^2_t$$

$$\text{tbi}_t = a^3_t + \sum_{i=1}^{4} b^3_{it}\ \text{inf}_{t-i} + \sum_{i=1}^{4} c^3_{it}\ \text{une}_{t-i} + \sum_{i=1}^{4} d^3_{it}\ texttbi_{t-i} + u^3_t$$

```
## 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 = 4, type = "const")
model.tvVAR <- tvVAR(usmacro, p = 4, type = "const")
```

The user can provide additional optional arguments to modify the default estimation. Refer to section 'User options of *tvAR*' to understand function parameters *p*, *z*, *bw*, *type*, *exogen*, *est*, *tkernel* and *singular.ok* for details. Note that the current version only allows one single random variable *z*, the same for every equation. The *tvVAR* model wraps estimator *tvOLS*. At the moment all equations are estimated as if there were independent, i.e. line by line. The variance-covariance matrix in the residuals can be used to calculate the orthogonal time-varying IRF function.

User options of *tvIRF*

The main parameters *x* which is an object of class *tvvar*. The user can provide additional optional arguments to modify the default estimation.

Impulse and response variables

Coefficients for every impulse and response variables are calculated by default. The user has the option to pick a subset of impulse variables and/or response variables using parameters *impulse* and *response*.

Horizon

The number of time periods ahead for which the impulse response function is calculated is 10. The user can input a longer of shorter horizon using parameter *n.ahead*.

Orthogonal tv-IRF

Function *tvIRF* calculates one impulse response function for each point in time. It is likely that the errors variance-covariance matrix of a process with time-varying coefficients is also time-varying. Those, the default of parameter *ortho.cov* is "tv". Note that the use can input a value of the bandwidth for the covariance matrix estimation in *bw.cov*. This function calls function *tvCov* in the calculation of the MA (∞) coefficients. It is possible to use a constant variance-covariance matrix by setting *ortho.cov* to “const”.

```
## Estimate a the impulse response functions with irf and tvIRF from previous
## vector autoregressive models
model.irf <- vars::irf(model.VAR)
model.tvIRF <- tvIRF(model.tvVAR)
```

Cumulative tv-IRF

The cumulative time-varying IRF is fitted when parameter *cumulative* is set to TRUE.

```
## Estimate a the tvIRF with time-varying covariance function The estimation
## bandwidth might be automatically calculated or input
model.tvIRF2 <- tvIRF(model.tvVAR, cumulative = TRUE)
```

Standard usage of the tvSURE

A time-varying coefficients SURE model is a multiequation where the regressors are exogenous variables. The main parameter of this function is a list of formulas, one for each equation. The formula format is inpired in the formula from package *systemfit* which estimates parametric multiequation problems with constant coefficients. The examples below use the dataset *Kmenta* from package *sysmtefit* to illustrate a demand/supply problem using different models.

The estimation of this multiequation problem can be done equation by equation, assuming no correlation in the error variance-covariance matrix, or as a system where the information in this matrix is used. Regressors can be different for each equation. An intercept is included by default in each equation unless its formula states it differently. Econometrically, function *tvOLS* is used by default, calculating estimates for each equation independently with different bandwidths *bw*. The user is able to input a set of bandwidths or a single bandwidth to be used in the estimation instead.

```
data("Kmenta", package = "systemfit")
eqDemand <- consump ~ price + income
eqSupply <- consump ~ price + farmPrice + trend
system <- list(demand = eqDemand, supply = eqSupply)

## OLS estimation of a system
ols.fit <- systemfit::systemfit(system, method = "OLS", data = Kmenta)

## tvOLS estimation of a system with the local linear estimator
tvols.fit <- tvSURE(system, data = Kmenta, est = "ll")
```

When the user desires to fit a time-varying coefficients SURE model, if the error variance-covariance matrix is known, parameter *method* is set to “tvGLS” and the variance-covariance matrix is input into parameter *Sigma*. Otherwise, if method is set to “tvGLS” and *Sigma* is NULL, the estimation is done as in the case method = “tvOLS”. When the user desires to fit a time-varying coefficients SURE model and Sigma is unknown, parameter *method* must be set to “tvFGLS”. The error variance-covariance matrix will be estimated with function *tvCov*.

```
## FGLS estimation - SURE estimation
fgls1.fit <- systemfit::systemfit(system, data = Kmenta, method = "SUR")

## tvFGLS estimation - tvSURE estimation
tvfgls1.fit <- tvSURE(system, data = Kmenta, method = "tvFGLS")
##'
```

```
## Iterative FGLS estimation - SUR estimation
fgls2.fit <- systemfit::systemfit(system, data = Kmenta, method = "SUR", maxit = 100)
```

User options of *tvSURE*

The user can provide additional optional arguments to modify the default estimation. Refer to section 'User options of *tvLM*' to understand function parameters *z*, *data*, *bw*, *est*, *tkernel* and *singular.ok* for details. Note that the current version only allows one single random variable *z*, the same one, for every equation. The *tvSURE* function wraps estimators *tvOLS* and *tvGLS*.

Choosing the estimation method

This parameter defines the type of estimation to be performed. The choices for parameter *method* are:

1. Item 1 "tvOLS" for a line by line estimation as if the error variance-covariance matrix, Σ was the identity matrix.
2. Item 2 "tvGLS" to estimate the coefficients of the system using Σ, for which the user must input it in parameter *Sigma*. Parameter *Sigma* takes either a symmetric matrix or an array. If Σ is considered constant, its number of rows and columns is equal to the number of equations (neq) in the system. If Σ is considered to chanve with time, then an array of dimension neq $\times$ neq $\times$ number of time periods must be input. Note that if the user inputs a diagonal variance-covariance matrix with diagonal values different from one, then a time-varying weighted least squares is performed. If "tvGLS" is chosen in *method* but *Sigma* is NULL, then method "tvOLS" is performed.

3. Item 3 "tvFGLS" to estimate the coefficients of the system using an estimated time-varying variance-covariance matrix. By default, only one iteration is performed in the estimation of Σ, unless *control* variable indicates otherwise. The user can choose the maximum number of iterations or the level of tolerance in the estimation of Σ. See example below for details.

```
## Iterative tvFGLS estimation - SURE estimation using the local linear
tvfgls2.fit <- tvSURE(system, data = Kmenta, method = "tvFGLS", control = list(tol = 0.001,
    maxiter = 100))
```

Coefficient restrictions

The user can restrict certain coefficients using parameters *R* and *r*. Note that the restriction is setting those coefficients to a constant. An illustration of this usage in the example below.

```
## Estimation with 2 restrictions
Rrestr <- matrix(0, 2, 7)
Rrestr[1, 3] <- 1
Rrestr[1, 7] <- -1
Rrestr[2, 2] <- -1
Rrestr[2, 5] <- 1
qrestr <- c(0, 0.5)
tvfgls.rest <- tvSURE(system, data = Kmenta, method = "tvFGLS", R = Rrestr,
    r = qrestr, bw = tvfgls1.fit$bw, bw.cov = tvfgls1.fit$bw.cov)
```

Estimating a time-varying variance-covariance matrix

A time-varying covariance matrix of two or more series can be estimated using function *tvCov* as in the example below. This is the function used by *tvVAR* and *tvSURE* to calculate a time-varying variance-covariance matrix of the error term.

```
library(MASS)
## Generate two independent (uncorrelated) series
y <- cbind(rnorm(200, sd = 4), rnorm(200, sd = 1))
## Calculate the bandwidth
bw.cov <- bwCov(y)
## Estimate variance-variance matrix
Sigma.hat <- tvCov(y, bw = bw.cov)
## The first time point estimate
print(Sigma.hat[, , 1])
```

```
##            [,1]       [,2]
## [1,] 14.9215396 -0.3108056
## [2,] -0.3108056  1.0024105
```

```
## The mean over time of all estimates
print(apply(Sigma.hat, 1:2, mean))
```

```
##            [,1]       [,2]
## [1,] 17.7053158 -0.0418074
## [2,] -0.0418074  1.1007584
```

```
## Generate two dependent variables with a covariance of -0.5
y <- mvrnorm(n = 200, mu = c(0, 0), Sigma = cbind(c(1, -0.5), c(-0.5, 4)))
## Calculate the bandwidth
bw.cov <- bwCov(y)
## Estimation the variables variance-covariance matrix
Sigma.hat <- tvCov(y, bw = bw.cov)
## The first time point estimate
print(Sigma.hat[, , 1])
```

```
##            [,1]       [,2]
## [1,]  1.1063227 -0.4436876
## [2,] -0.4436876  4.5288380
```

Confidence intervals

Package *tvReg* calculates the confidence intervals of objects of class *tvlm*, *tvar*, *tvsure* and *tvirf* with function *CI*. Parameter *level* is set to 0.95 (95% confidence interval) by default. If this parameter is chosen to be a number between 0 and 1, then a number of *runs* resamples is used in the confidence interval calculation. Because coefficients are time-varying, only wild bootstrap residual resampling is possible. Two choices of wildbootstrap are available in parameter *tboot*, the Mammen (1993) (default) and the one using the normal distribution (tboot =“wild2”).

Note that if the input model in function *CI* is the same than the output model, the calculation of following confidence intervals will be faster as the results from replications are stored in variable *BOOT*.

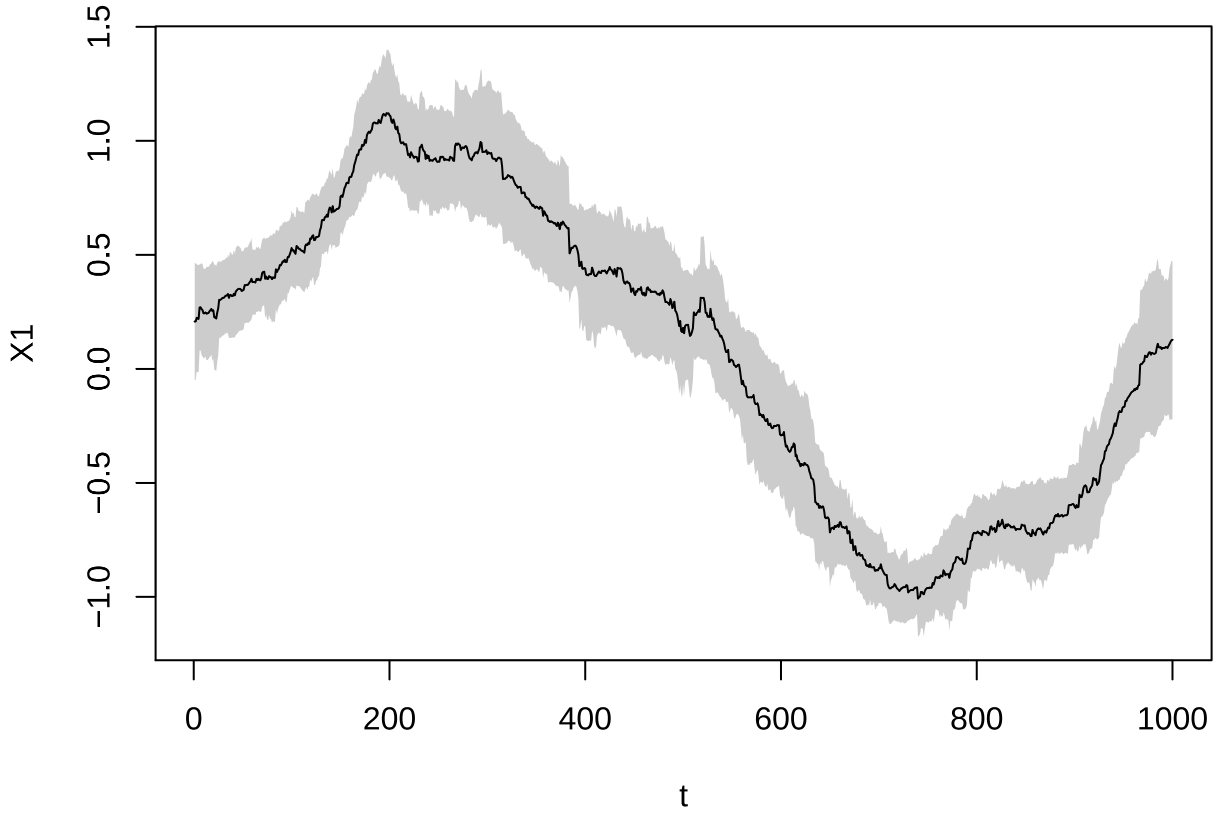
```
## Obtain the 90% confidence interval of the coefficients for an object of
## class tvlm
model.tvlm.90 <- CI(model.tvlm, level = 0.9, runs = 50)
```

```r
## Obtain the 95% confidence interval of the same object. This will reused
## the resamples of object model.tvlm done previously. So the second
## confidence interval calculation is faster
model.tvlm.95 <- CI(model.tvlm.90)

## 80% confidence interval using normal wild bootstrap for object of class
## tvar with 200 bootstrap resamples
model.tvAR.ci <- CI(model.tvAR.1p.z, tboot = "wild2", level = 0.8, runs = 50)

## 95% confidence interval using Mammen's wild bootstrap for object of class
## tvirf
model.tvIRF <- CI(model.tvIRF)

## 80% confidence interval using Mammen's wild bootstrap for a cummulatife
## tvirf
model.tvIRF2 <- CI(model.tvIRF2, level = 0.8)

## 50% confidence interval using Mammen's wild bootstrap for object of class
## tvsure
tvols.fit <- CI(tvols.fit, level = 0.5)
```

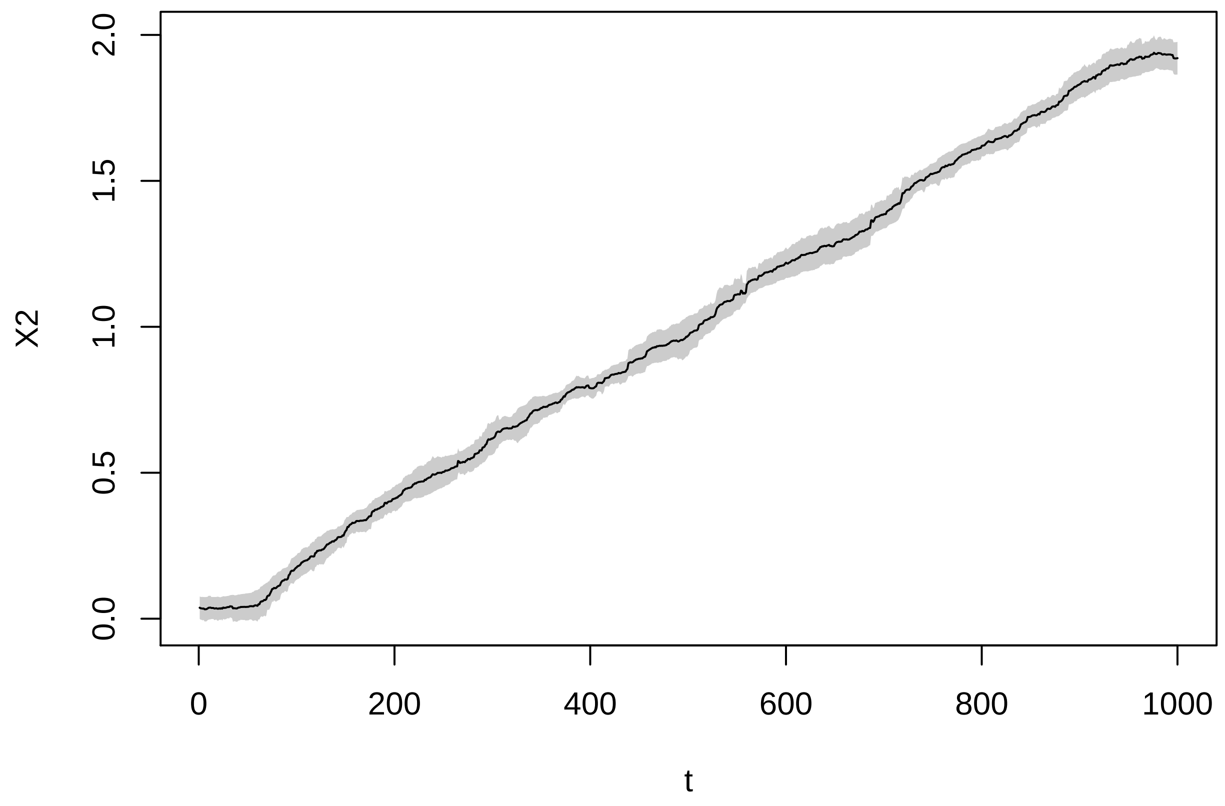
Plot

There is a plot function for objects of the five classes in this package: *tvlm*, *tvar*, *tvvar*, *tvsure* and *tvirf*. The variable coefficients estimates, as well as if confidence intervals if calculated, are plotted for classes *tvlm*, *tvar*, *tvsure* and *tvirf*. The fitted values and the residuals are plotted for class *tvvar* because the interpretation of its coefficients is not relevant.

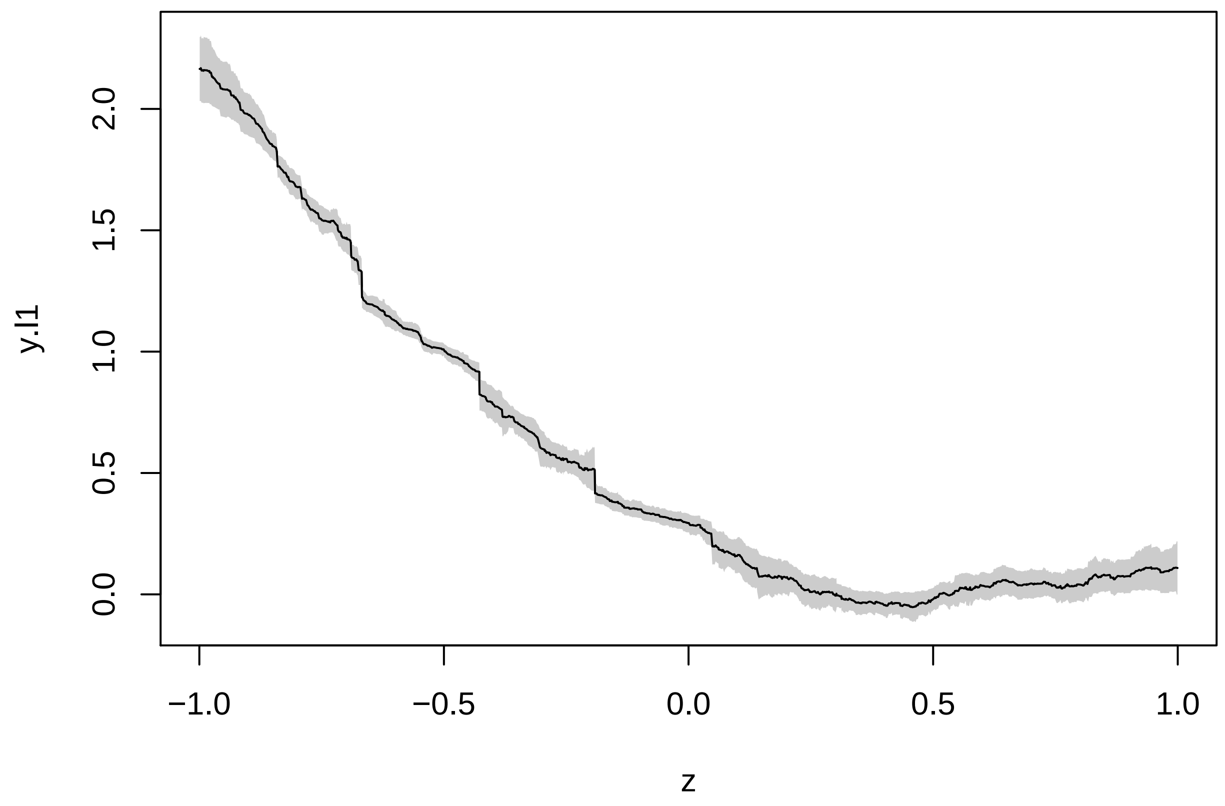
```r
## Plot coefficient estimates and confidence intervals (if calculated) of
## objects of class tvlm
plot(model.tvlm.90)
```



X1
1.5
1.0
0.5
0.0
−0.5
−1.0
0
200
400
600
800
1000
t



```
## Plot coefficient estimates of objects of class tvar.
plot(model.tvAR.ci)
```



```r
## Plot the fitted values and residuals of each equation in the model
plot(model.tvVAR)
```

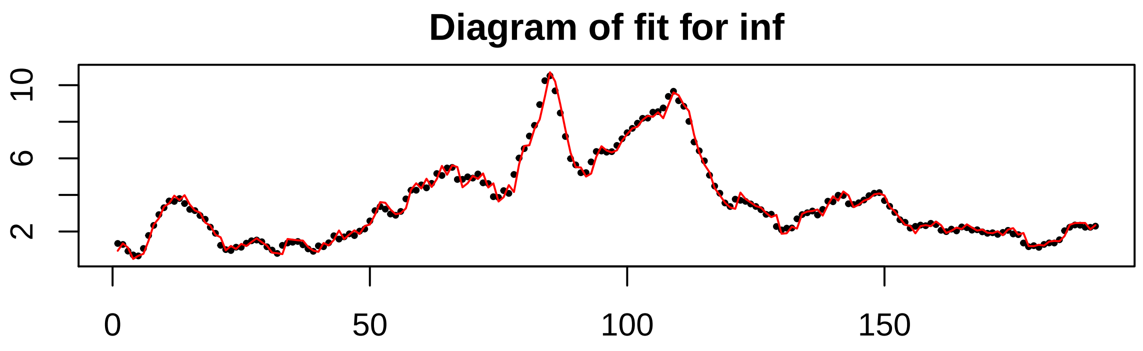


Diagram of fit for inf
2
6
10
0
50
100
150

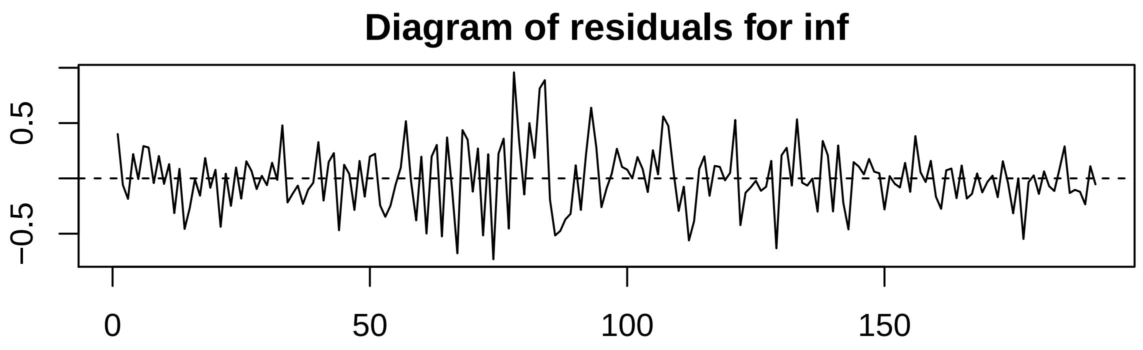


Diagram of residuals for inf
−0.5
0.5
0
50
100
150

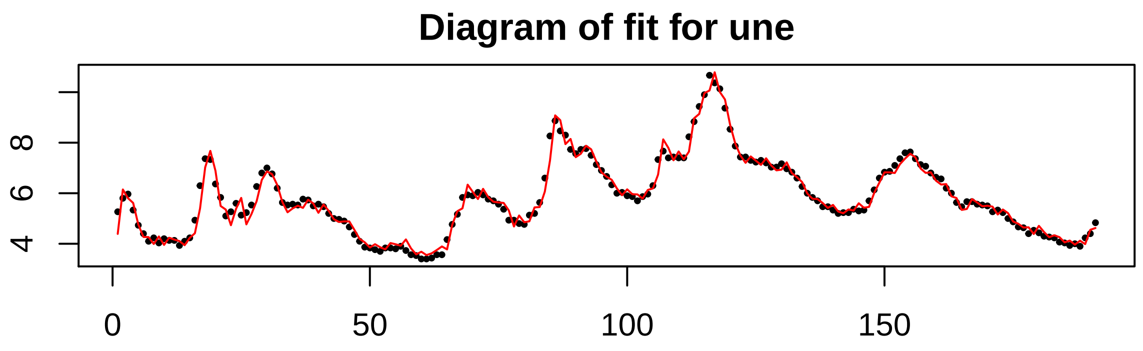


Diagram of fit for une
4
6
8
0
50
100
150

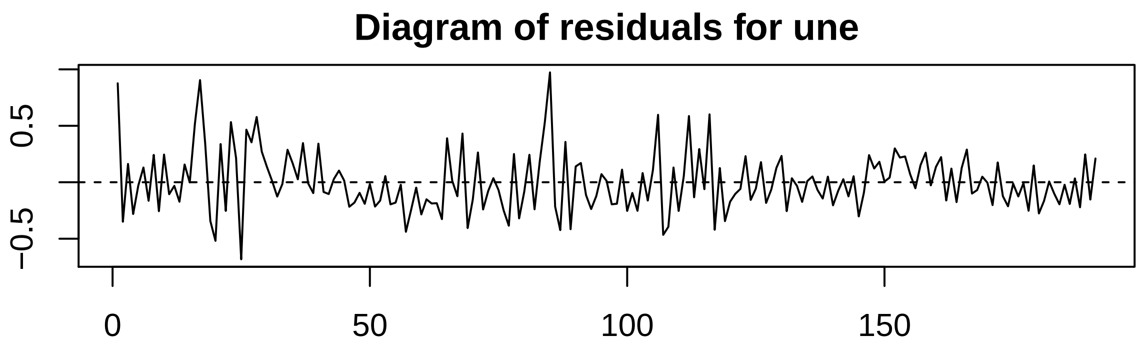
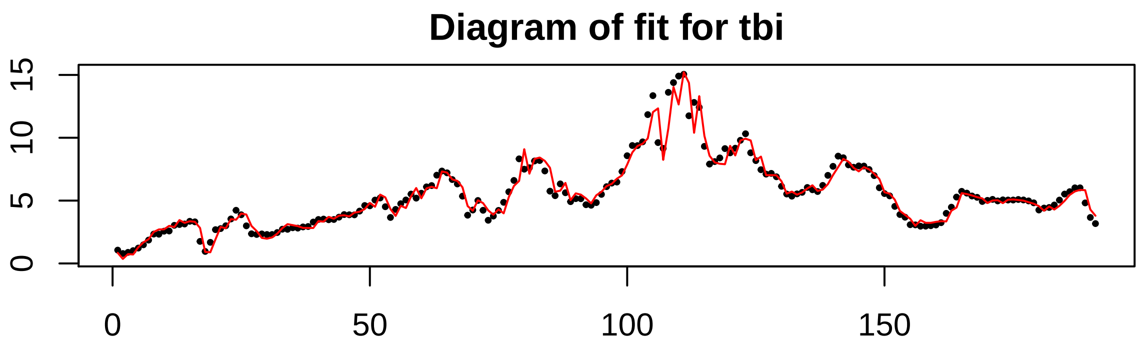
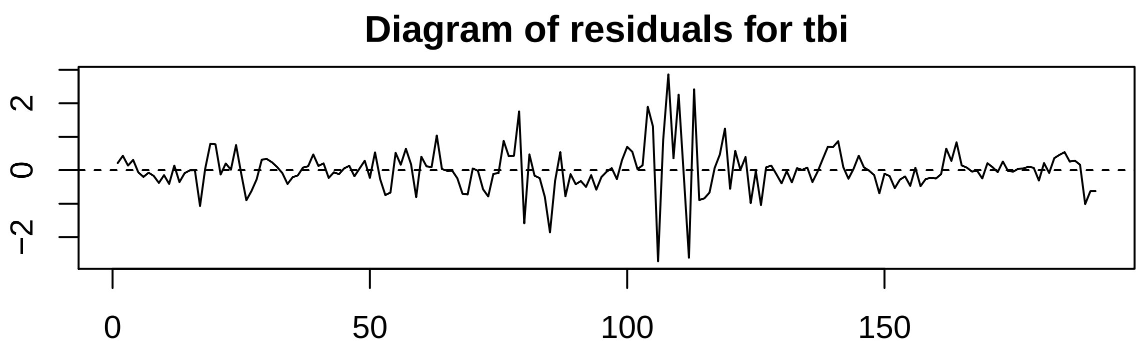


Diagram of residuals for une
−0.5
0.5
0
50
100
150

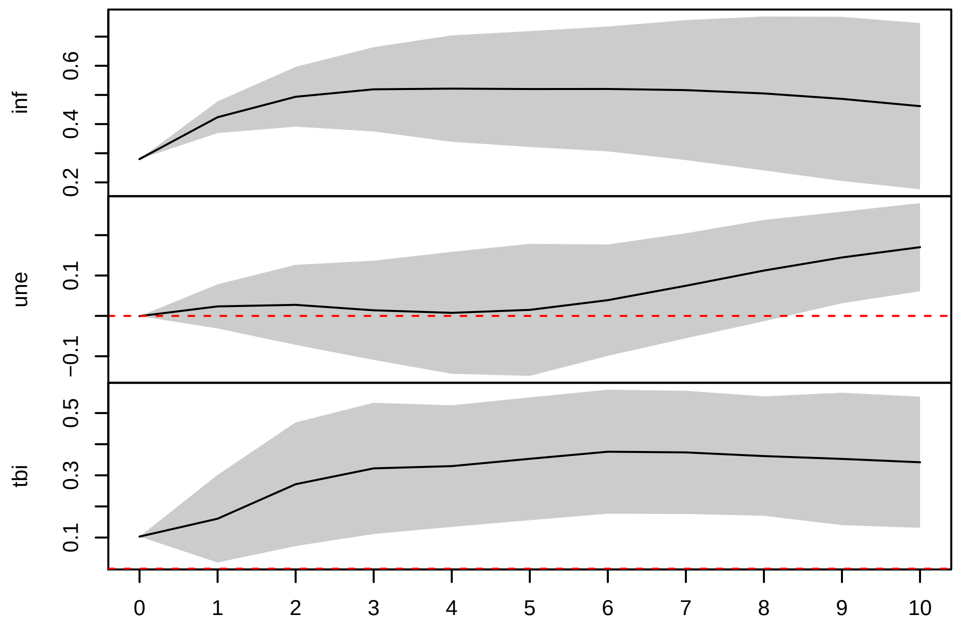




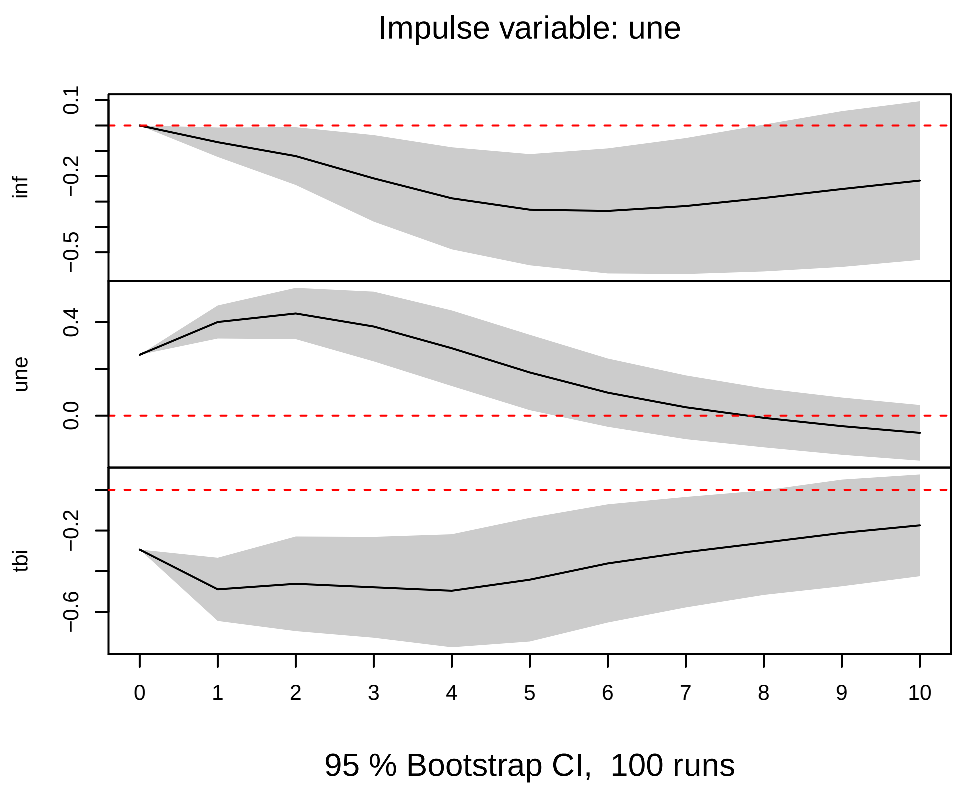
```r
## Plot the mean all tvIRF over time
plot(model.tvIRF)
```

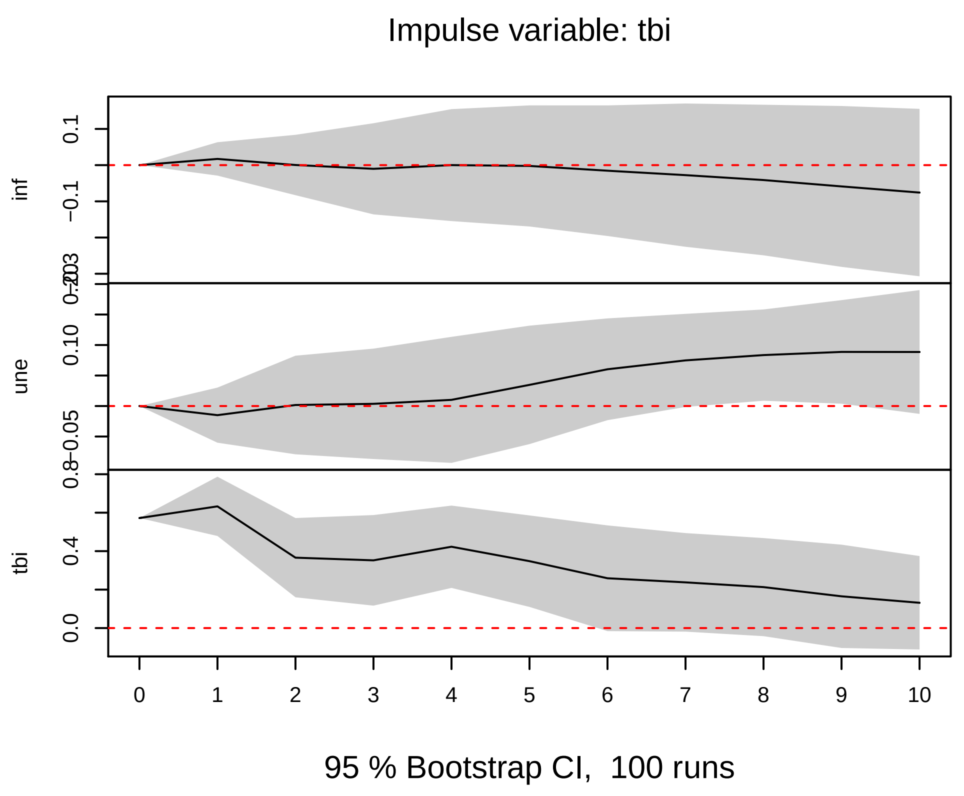
```
##
## The plot represents the mean of tvIRF over every time period. Enter a row number in obs.index
##          to plot the tvIRF of a particular point in time.
```

Impulse variable: inf

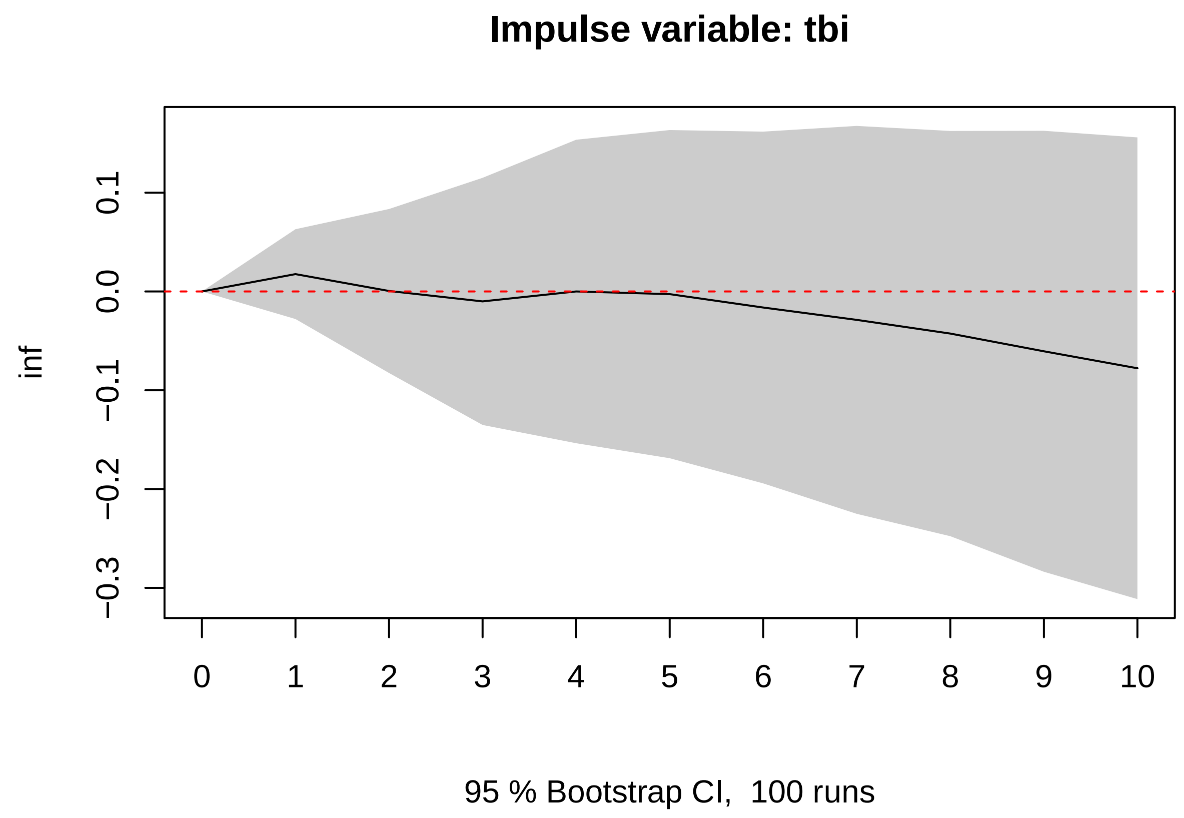


95 % Bootstrap CI, 100 runs


Impulse variable: une
inf
une
tbi
0.1
−0.2
−0.5
0.4
0.0
−0.2
−0.6
0
1
2
3
4
5
6
7
8
9
10
95 % Bootstrap CI, 100 runs

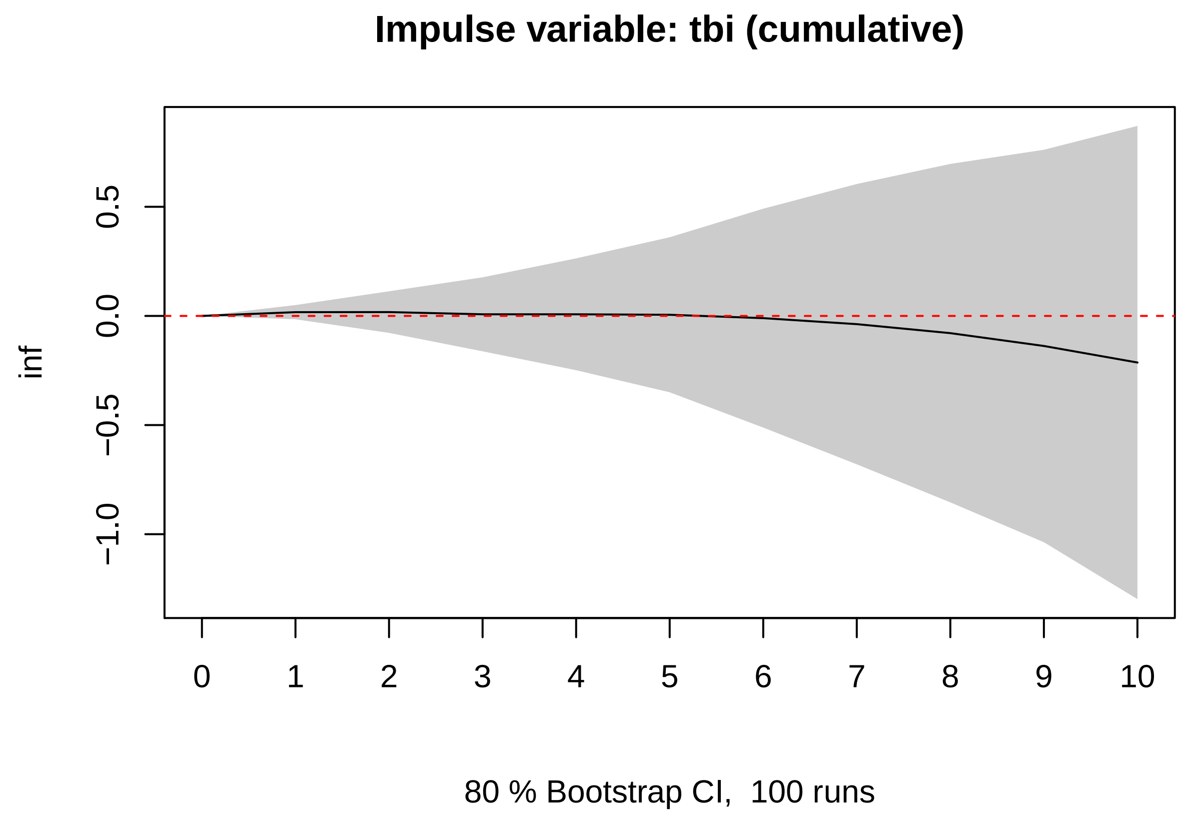


```
## Plot the effect of a shock in the interest rates (tbi) on the inflation
## (inf) at time 100
plot(model.tvIRF, obs.index = 100, impulse = "tbi", response = "inf")
```

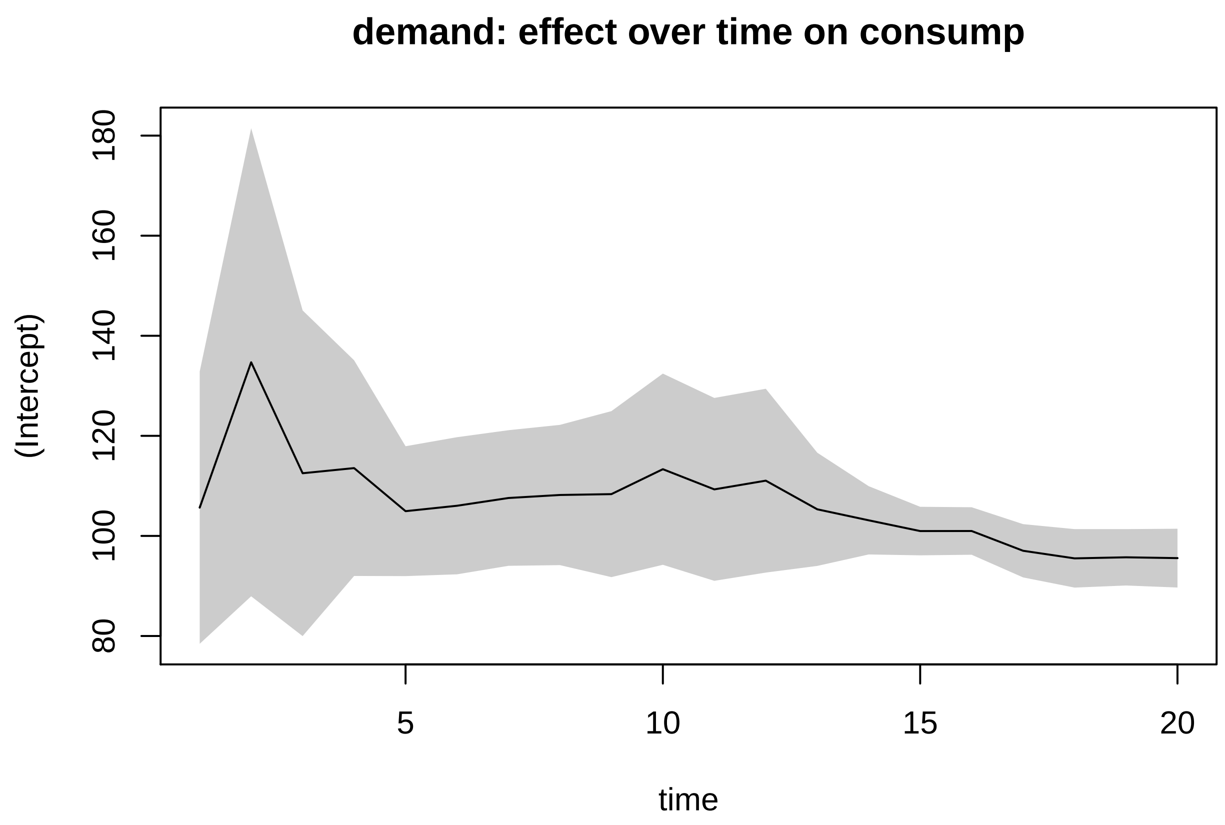


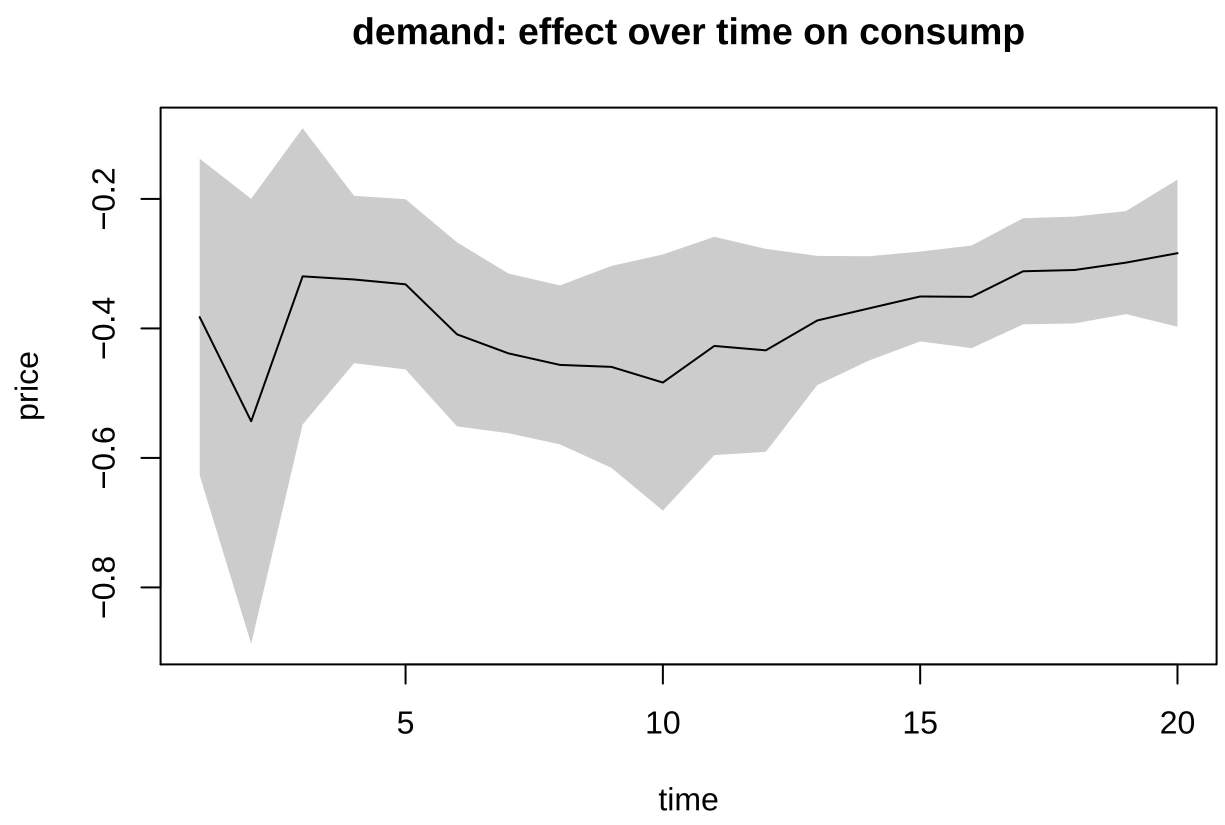
```
## Plot the cumulative effect on a shock in short term interest rates (tbi)
## on the inflation (inf)
plot <- plot(model.tvIRF2, impulse = "tbi", response = "inf")
```

```
##
## The plot represents the mean of tvIRF over every time period. Enter a row number in obs.index
##          to plot the tvIRF of a particular point in time.
```

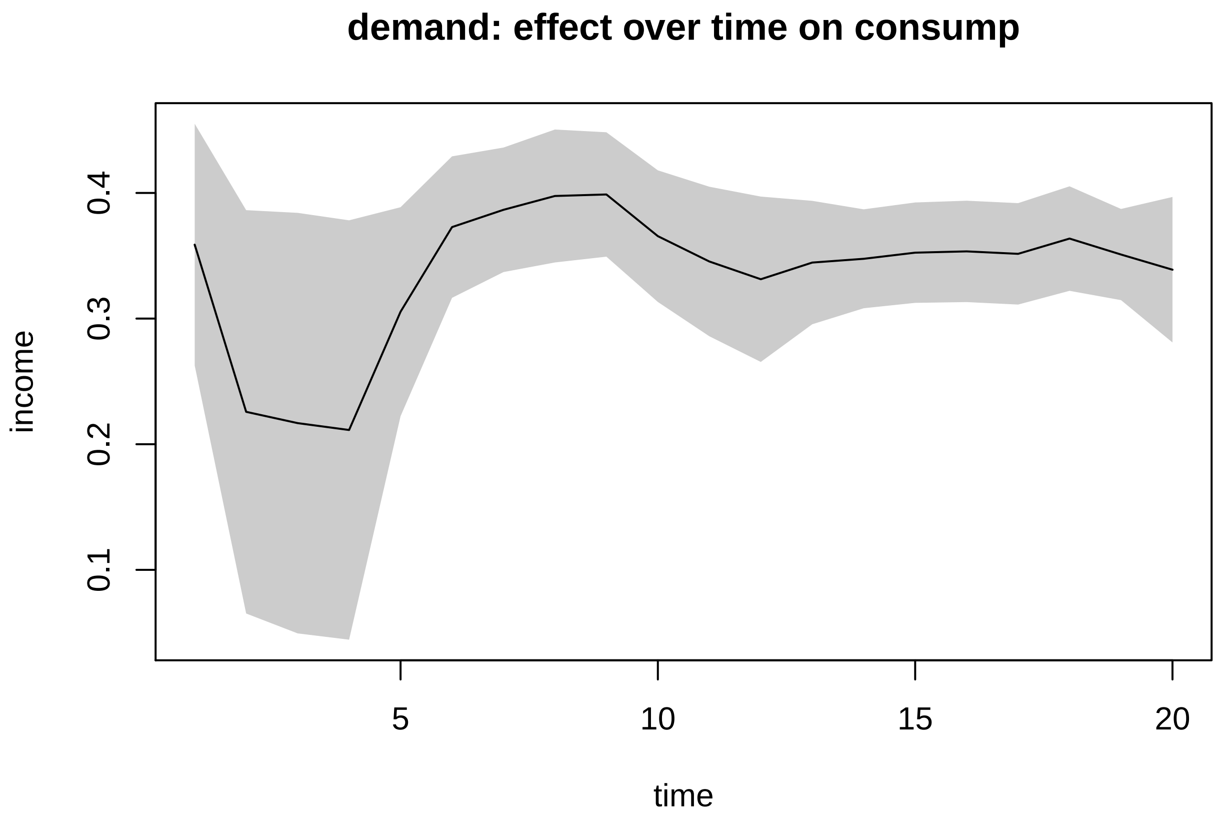


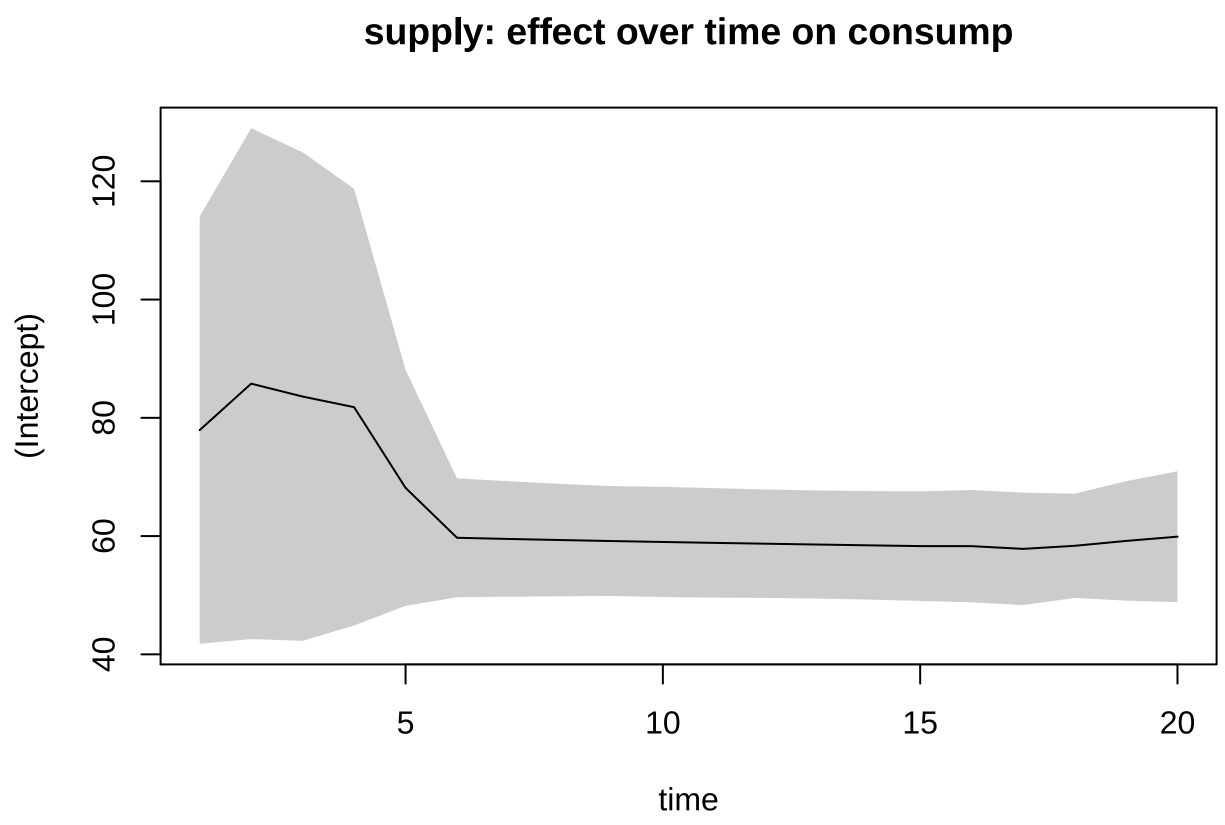
```
## Plot coefficient estimates and confidence intervals (if calculated) of
## objects of class tvsure
plot(tvols.fit)
```

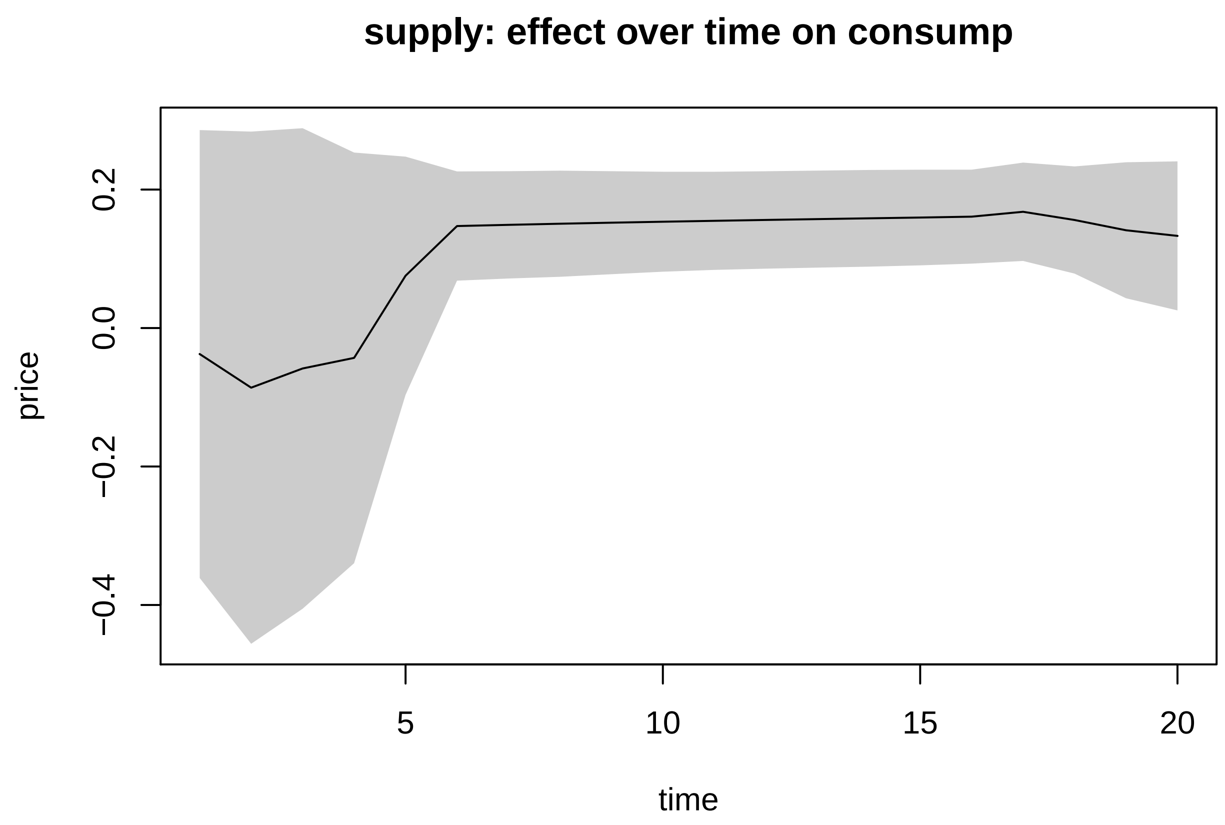

demand: effect over time on consump
(Intercept)
80
100
120
140
160
180
5
10
15
20
time



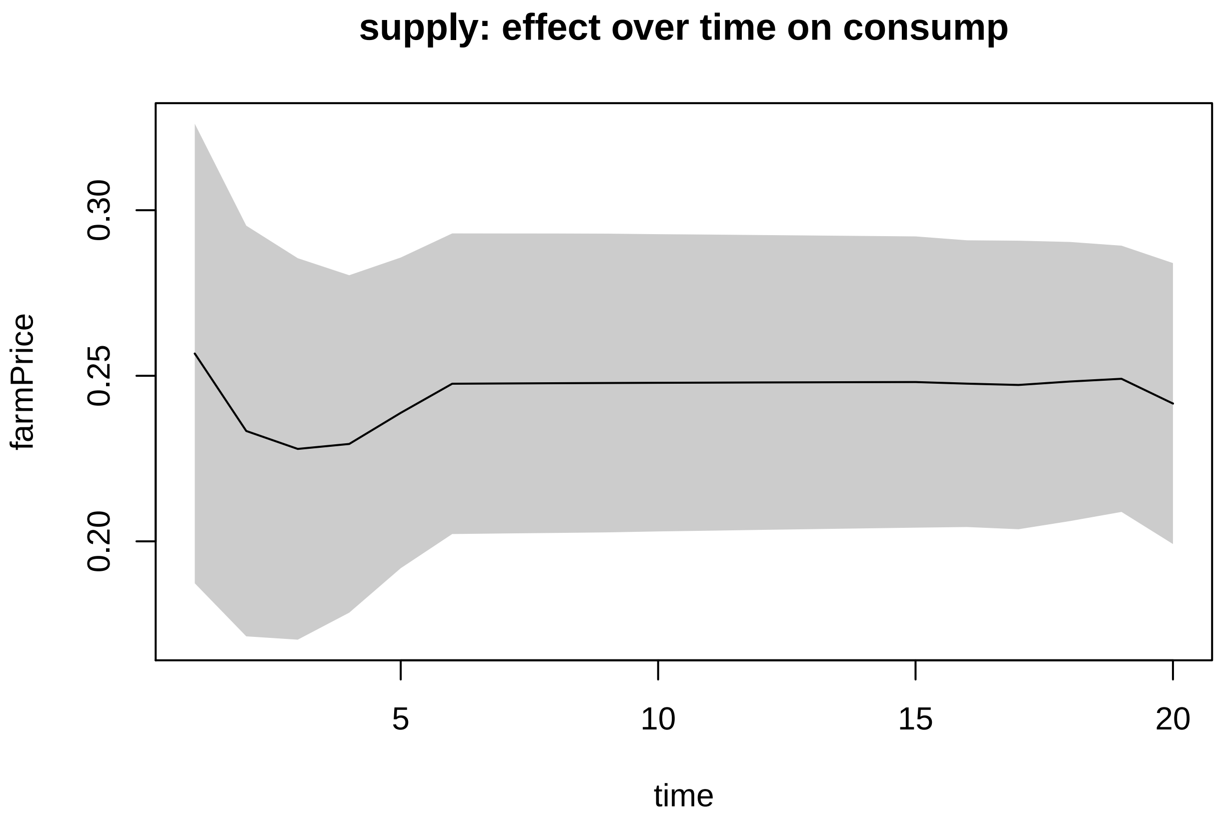
demand: effect over time on consump
price
−0.2
−0.4
−0.6
−0.8
5
10
15
20
time


demand: effect over time on consump
income
0.1
0.2
0.3
0.4
time
5
10
15
20


supply: effect over time on consump
(Intercept)
40
60
80
100
120
5
10
15
20
time



supply: effect over time on consump
price
0.2
0.0
−0.2
−0.4
5
10
15
20
time


supply: effect over time on consump
farmPrice
0.20
0.25
0.30
5
10
15
20
time