Package ‘gmm’

March 15, 2018

Version 1.6-2
Date 2017-09-26
Title Generalized Method of Moments and Generalized Empirical Likelihood
Author Pierre Chausse <pchausse@uwaterloo.ca>
Maintainer Pierre Chausse <pchausse@uwaterloo.ca>
Depends R (>= 2.10.0), sandwich
Suggests mvtnorm, car, stabedist, MASS, timeDate, timeSeries
Imports stats, methods, grDevices, graphics
License GPL (>= 2)
Repository CRAN
Repository/R-Forge/Project gmm
Repository/R-Forge/Revision 119
Repository/R-Forge/DateTimeStamp 2018-03-12 14:25:26
Date/Publication 2018-03-15 22:51:20 UTC
NeedsCompilation no

R topics documented:

ATEgel ......................................................... 2
bread ............................................................ 5
bwWilhelm ....................................................... 6
ATEgel

ATE with Generalized Empirical Likelihood estimation

Description

Function to estimate the average treatment effect with the sample being balanced by GEL.

Usage

ATEgel(g, balm, w=NULL, y=NULL, treat=NULL, tet@=NULL, momType=c("bal", "balSample", "ATT"),
popMom = NULL, family=c("linear", "logit", "probit"),
type = c("EL", "ET", "CUE", "ETEL", "HD", "ETHD"),
tol_lam = 1e-9, tol_obj = 1e-9, tol_mom = 1e-9, maxiterlam = 100,
optfct = c("optim", "nlminb"),
optlam = c("nlminb", "optim", "iter", "Wu"), data=NULL,
Lambdacontrol = list(),
model = TRUE, X = FALSE, Y = FALSE, ...)

checkConv(obj, tolConv=1e-4, verbose=TRUE)
Arguments

\( g \)  
A formula as \( y \sim z \), where \( codey \) is the response and \( z \) the treatment indicator. If there is more than one treatment, more indicators can be added or \( z \) can be set as a factor. It can also be of the form \( g(\theta, y, z) \) for non-linear models. It is however, not implemented yet.

\( obj \)  
Object of class "ategel" produced \( ATEgel \)

\( balm \)  
A formula for the moments to be balanced between the treated and control groups (see details)

\( y \)  
The response variable when \( g \) is a function. Not implemented yet

\( treat \)  
The treatment indicator when \( g \) is a function. Not implemented yet

\( w \)  
A formula to add covariates to the main regression. When NULL, the default value, the main regression only include treatment indicators.

\( tet0 \)  
A 3 \( \times \) 1 vector of starting values. If not provided, they are obtained using an OLS regression

\( momType \)  
How the moments of the covariates should be balanced. By default, it is simply balanced without restriction. Alternatively, moments can be set equal to the sample moments of the whole sample, or to the sample moments of the treated group. The later will produce the average treatment effect of the treated (ATT)

\( popMom \)  
A vector of population moments to use for balancing. It can be used of those moments are available from a census, for example. When available, it greatly improves efficiency.

\( family \)  
By default, the outcome is linearly related to the treatment indicators. If the outcome is binary, it is possible to use the estimating equations of either the logit or probit model.

\( type \)  

\( tol_lam \)  
Tolerance for \( \lambda \) between two iterations. The algorithm stops when \( \| \lambda_i - \lambda_{i-1} \| \) reaches \( tol_lam \) (see \( getLamb \))

\( maxiterlam \)  
The algorithm to compute \( \lambda \) stops if there is no convergence after "maxiterlam" iterations (see \( getLamb \)).

\( tol_obj \)  
Tolerance for the gradient of the objective function to compute \( \lambda \) (see \( getLamb \)).

\( optfct \)  
Algorithm used for the parameter estimates

\( tol_mom \)  
It is the tolerance for the moment condition \( \sum_{i=1}^{n} p_i g(\theta(x_i)) = 0 \), where \( p_i = \frac{1}{n} D \rho(< g_i, \lambda >) \) is the implied probability. It adds a penalty if the solution diverges from its goal.

\( optlam \)  
Algorithm used to solve for the lagrange multiplier in \( getLamb \). The algorithm Wu is only for type="EL".

\( data \)  
A data.frame or a matrix with column names (Optional).

\( Lambdacontrol \)  
Controls for the optimization of the vector of Lagrange multipliers used by either \( optim, nlminb \) or \( constrOptim \)
model, X, Y  logicals. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response) are returned if g is a formula.

verbose If TRUE, a summary of the convergence is printed

tolConv The tolerance for comparing moments between groups

Details

We want to estimate the model \( Y_t = \theta_1 + \theta_2 \text{treat} + \epsilon_t \), where \( \theta_2 \) is the treatment effect. GEL is used to balance the sample based on the argument x above.

For example, if we want the sample mean of \( x_1 \) and \( x_2 \) to be balanced between the treated and control, we set \( x \) to \( \sim x_1 + x_2 \). If we want the sample mean of \( x_1, x_2, x_1^2, x_2^2, x_1^2 + 2x_1x_2, x_2^2 + 2x_2 \), we set \( x \) to \( \sim x_1 + x_2 + I(x_1^2) + I(x_2^2) \).

Value

'gel' returns an object of 'class' 'ategel'

The functions 'summary' is used to obtain and print a summary of the results.

The object of class "ategel" is a list containing the same elements contained in objects of class gel.

References


Examples

data(nsw)
# Scale income
nsw$re78 <- nsw$re78/1000
nsw$re75 <- nsw$re75/1000
res <- ATEgel(re78~treat, ~age+ed+black+hisp+married+nodeg+re75, data=nsw,type="ET")
summary(res)
chk <- checkConv(res)
res2 <- ATEgel(re78~treat, ~age+ed+black+hisp+married+nodeg+re75, data=nsw,type="ET", momType="balSample")
bread

summary(res2)
chk2 <- checkConv(res2)

bread

Bread for sandwiches

Description

Computes the bread of the sandwich covariance matrix

Usage

## S3 method for class 'gmm'
bread(x, ...)

## S3 method for class 'gel'
bread(x, ...)

## S3 method for class 'tls'
bread(x, ...)

Arguments

x A fitted model of class gmm or gel.

... Other arguments when bread is applied to another class object

Details

When the weighting matrix is not the optimal one, the covariance matrix of the estimated coefficients is:

\[
(G'WG)^{-1}G'WVWG(G'WG)^{-1}
\]

where \(G = \frac{dg}{d\theta}\), \(W\) is the matrix of weights, and \(V\) is the covariance matrix of the moment function. Therefore, the bread is \((G'WG)^{-1}\), which is the second derivative of the objective function.

The method if not yet available for gel objects.

Value

A \(k \times k\) matrix (see details).

References

Examples

# See \code{\link{gmm}} for more details on this example.
# With the identity matrix
# bread is the inverse of (G'G)

n <- 1000
x <- rnorm(n, mean = 4, sd = 2)
g <- function(tet, x)
{
  m1 <- (tet[1] - x)
  m2 <- (tet[2]^2 - (x - tet[1])^2)
  m3 <- x^3 - tet[1]*(tet[1]^2 + 3*tet[2]^2)
  f <- cbind(m1, m2, m3)
  return(f)
}

Dg <- function(tet, x)
{
  return(jacobian)
}

res <- gmm(g, x, c(0, 0), grad = Dg, weightsMatrix=diag(3))
G <- Dg(res$coeff, x)
bread(res)
solve(crossprod(G))

bwWilhelm

Wilhelm (2015) bandwidth selection

Description

It computes the optimal bandwidth for the HAC estimation of the covariance matrix of the moment conditions. The bandwidth was shown by Wilhelm (2005) to be the one that minimizes the MSE of the GMM estimator.

Usage

bwWilhelm(x, order.by = NULL, kernel = c("Quadratic Spectral", "Bartlett", "Parzen", "Tukey-Hanning"), approx = c("AR(1)", "ARMA(1,1)").weights = NULL, prewhite = 1, ar.method = "ols", data = list())

Arguments

x
An object of class gmm.

order.by
Either a vector ‘z’ or a formula with a single explanatory variable like ‘~ z’. The observations in the model are ordered by the size of ‘z’. If set to ‘NULL’ (the default) the observations are assumed to be ordered (e.g., a time series).
kernel  type of kernel used to compute the covariance matrix of the vector of sample moment conditions (see kernHAC for more details)

approx A character specifying the approximation method if the bandwidth has to be chosen by bwAndrews.

weights numeric. A vector of weights used for weighting the estimated coefficients of the approximation model (as specified by ‘approx’). By default all weights are 1 except that for the intercept term (if there is more than one variable)

prewhite logical or integer. Should the estimating functions be prewhitened? If TRUE or greater than 0 a VAR model of order as.integer(prewhite) is fitted via ar with method “ols” and demean = FALSE.

ar.method character. The method argument passed to ar for prewhitening.

data an optional data frame containing the variables in the ‘order.by’ model.

Value

The function ‘bwWilhelm’ returns the optimal bandwidth.

Note

The function was written by Daniel Wilhelm and is based on bwAndrews.

References


Examples

data(Finance)
f1 <- Finance[1:300, "rm"]
f2 <- Finance[1:300, "hml"]
f3 <- Finance[1:300, "smb"]
y <- Finance[1:300,"Wmk"]

## Silly example just to make it over-identified
res <- gmm(y ~ f1 - f1 + f2 + f3)
summary(res)

## Set the bandwidth using the second step estimate
bw <- bwWilhelm(res)
res2 <- update(res, bw=bw)
summary(res2)

## Set the bandwidth using the first-step estimate as for bwAndrews
The characteristic function of a stable distribution

Description

It computes the theoretical characteristic function of a stable distribution for two different parametrizations. It is used in the vignette to illustrate the estimation of the parameters using GMM.

Usage

charStable(theta, tau, pm = 0)

Arguments

theta     Vector of parameters of the stable distribution. See details.
tau      A vector of numbers at which the function is evaluated.
    pm    The type of parametization. It takes the values 0 or 1.

Details

The function returns the vector $\Psi(\theta, \tau, pm)$ defined as $E(e^{ix\tau})$, where $\tau$ is a vector of real numbers, $i$ is the imaginary number, $x$ is a stable random variable with parameters $\theta = (\alpha, \beta, \gamma, \delta)$ and pm is the type of parametrization. The vector of parameters are the characteristic exponent, the skewness, the scale and the location parameters, respectively. The restrictions on the parameters are: $\alpha \in (0, 2]$, $\beta \in [-1, 1]$ and $\gamma > 0$. For mode details see Nolan(2009).

Value

It returns a vector of complex numbers with the dimension equals to length(tau).

References

Examples

# GMM is like GLS for linear models without endogeneity problems

pm <- 0
theta <- c(1.5,5,1,0)
tau <- seq(-3, 3, length.out = 20)
char_fct <- charStable(theta, tau, pm)

cdf <- coef

Description

It extracts the coefficients from gel or gmm objects.

Usage

## S3 method for class 'gmm'
coef(object, ...)

## S3 method for class 'gel'
coef(object, lambda = FALSE, ...)

Arguments

object An object of class gel or gmm returned by the function gel or gmm
lambda If set to TRUE, the lagrange multipliers are extracted instead of the vector of coefficients
...
Other arguments when coef is applied to an other class object

Value

Vector of coefficients

Examples

####################################
n = 500
phi <- c(.2, .7)
theta <- 0
sd <- .2
x <- matrix(arima.sim(n=n, list(order=c(2,0,1), ar=phi, ma=theta, sd=sd)), ncol=1)
y <- x[7:n]
yml1 <- x[6:(n-1)]
ynm2 <- x[5:(n-2)]

coef

Coefficients of GEL or GMM
### Description

It produces confidence intervals for the coefficients from `gel` or `gmm` estimation.

### Usage

```r
## S3 method for class 'gel'
confint(object, parm, level = 0.95, lambda = FALSE,
    type = c("Wald", "invLR", "invLM", "invJ"),
    fact = 3, corr = NULL, ...)

## S3 method for class 'gmm'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'ategel'
confint(object, parm, level = 0.95, lambda = FALSE,
    type = c("Wald", "invLR", "invLM", "invJ"), fact = 3,
    corr = NULL, robToMiss=TRUE, ...)

## S3 method for class 'confint'
print(x, digits = 5, ...)
```

### Arguments

- **object**
  - An object of class `gel` or `gmm` returned by the function `gel` or `gmm`
- **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 considered.
- **level**
  - The confidence level
- **lambda**
  - If set to TRUE, the confidence intervals for the Lagrange multipliers are produced.
- **type**
  - 'Wald' is the usual symmetric confidence interval. The others are based on the inversion of the LR, LM, and J tests.
This parameter controls the span of search for the inversion of the test. By default we search within plus or minus 3 times the standard error of the coefficient estimate.

This numeric scalar is meant to apply a correction to the critical value, such as a Bartlett correction. This value depends on the model (See Owen; 2001).

An object of class confint produced by confint.gel and confint.gmm

The number of digits to be printed

If TRUE, the confidence interval is based on the standard errors that are robust to misspecification

Other arguments when confint is applied to another class object

Value

It returns a matrix with the first column being the lower bound and the second the upper bound.

References


Examples

```
########
n = 500
phi <- c(2, .7)
the <- 0
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2,0,1), ar = phi, ma = the, sd = sd)), ncol = 1)
y <- x[7:n]
ym1 <- x[6:(n-1)]
ym2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y ~ ym1 + ym2
x <- H
t0 <- c(0, .5, .5)
resGel <- gel(g, x, t0)
confint(resGel)
confint(resGel, level = 0.90)
confint(resGel, lambda = TRUE)

########
resGmm <- gmm(g, x)
```
estfun

Extracts the empirical moment function

Description

It extracts the matrix of empirical moments so that it can be used by the kernHAC function.

Usage

```
# S3 method for class 'gmmFct'
estfun(x, y = NULL, theta = NULL, ...)
# S3 method for class 'gmm'
estfun(x, ...)
# S3 method for class 'gel'
estfun(x, ...)
# S3 method for class 'tsls'
estfun(x, ...)
```

Arguments

- `x`  
  A function of the form \( g(\theta, y) \) or a \( n \times q \) matrix with typical element \( g_i(\theta, y_t) \) for \( i = 1, \ldots, q \) and \( t = 1, \ldots, n \) or an object of class `gmm`. See `gmm` for more details. For `tsls`, it is an object of class `tsls`.

## Example

```r
set.seed(11233)
x <- rt(40, 3)
y <- x + rt(40, 3)
# Simple interval on the mean
res <- gel(x=1, -1, method = "Brent", lower = -4, upper = 4)
confint(res, type = "invLR")
confint(res)
# Using a Bartlett correction
k <- mean((x - mean(x))^4)/sd(x)^4
s <- mean((x - mean(x))^3)/sd(x)^3
a <- k/2 - s^2/3
corr <- 1+a/40
confint(res, type = "invLR", corr = corr)

# Interval on the slope
res <- gel(y ~ x, -x)
confint(res, "x", type = "invLR")
confint(res, "x")
```
**estfun**

Object

An object of class `tsls`.

**y**

The matrix or vector of data from which the function \( g(\theta, y) \) is computed if \( g \) is a function.

**theta**

Vector of parameters if \( g \) is a function.

**...**

Other arguments when `estfun` is applied to another class object.

Details

For `estfun.gmmFct`, it returns a \( n \times q \) matrix with typical element \( g_i(\theta, y_t) \) for \( i = 1, \ldots, q \) and \( t = 1, \ldots, n \). It is usually used by `gmm` to obtain the estimates.

For `estfun.gmm`, it returns the matrix of first order conditions of \( \min_\theta \bar{g}'W\bar{g}/2 \), which is a \( n \times k \) matrix with the \( t^{th} \) row being \( g(\theta, y_t)WG \), where \( G \) is \( d\bar{g}/d\theta \). It allows to compute the sandwich covariance matrix using `kernHAC` or `vcovHAC` when \( W \) is not the optimal matrix.

The method is not yet available for `gel` objects.

For `tsls`, `model.matrix` and `estfun` are used by `vcov()` to compute different covariance matrices using the `sandwich` package. See `vcov.tsls`. `model.matrix` returns the fitted values from the first stage regression and `esfun` the residuals.

Value

A \( n \times q \) matrix (see details).

References


Examples

```r
n = 500
phi <- c(.2, .7)
theta <- 0
sd <- .2
x <- matrix(arima.sim(n=n, list(order=c(2,0,1), ar=phi, ma=theta, sd=sd)), ncol=1)
y <- x[7:n]
yml <- x[6:(n-1)]
ym2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y ~ ym1 + ym2
x <- H
res <- gmm(g, x, weightsMatrix = diag(5))
gt <- res$gt
G <- res$G

foc <- gt
foc2 <- estfun(res)
```
foc[1:5,]
foc2[1:5,]

## Description

Daily returns on selected stocks, the Market portfolio and factors of Fama and French from 1993-01-05 to 2009-01-30 for CAPM and APT analysis

## Usage

data(Finance)

## Format

A data frame containing 24 time series. Dates are reported as rownames(). In the following description, company symbols are used.

- **WMK** Returns of WEIS MARKETS INC
- **UIS** Returns of UNISYS CP NEW
- **ORB** Returns of ORBITAL SCIENCES CP
- **MAT** Returns of Mattel, Inc.
- **ABAX** Returns of ABAXIS, Inc.
- **T** Returns of AT&T INC.
- **EMR** Returns of EMERSON ELEC CO
- **JCS** Returns of Communications Systems Inc.
- **VOXX** Returns of Audiovox Corp.
- **ZOOM** Returns of ZOOM Technologies Inc.
- **TDW** Returns of TIDEWATER INC
- **ROG** Returns of Rogers Corporation
- **GGG** Returns of Graco Inc.
- **PC** Returns of Panasonic Corporation
- **GCO** Returns of Genesco Inc.
- **EBF** Returns of ENNIS, INC
- **F** Returns of FORD MOTOR CO
- **FNM** Returns of FANNIE MAE
- **NHP** Returns of NATIONWIDE HLTH PROP
- **AA** Returns of ALCOA INC
- **rf** Risk-free rate of Fama-French
- **rm** Return of the market portfolio of Fama-French
- **hml** Factor High-Minus-Low of Fama-French
- **smb** Factor Small-Minus-Big of Fama-French
**FinRes**

*Method to finalize the result of the momentEstim method*

---

**Description**

It computes the final results that will be needed to create the object of class `gmm`.

**Usage**

```r
## S3 method for class 'baseGmm.res'
FinRes(z, object, ...)
```

**Arguments**

- `z`: An object of class determined by the method `momentEstim`.
- `object`: An object produced my `getModel`.
- `...`: Other argument to be passed to other `FinRes` methods.

**Value**

It returns an object of class `gmm`. See `gmm` for more details.

**References**


---

**fitted**

*Fitted values of GEL and GMM*

---

**Description**

Method to extract the fitted values of the model estimated by `gel` or `gmm`.

**Usage**

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

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

object     An object of class gel or gmm returned by the function gel or gmm

...     Other arguments when fitted is applied to an other class object

Value

It returns a matrix of the estimated mean $\hat{y}$ in $g=y \sim x$ as it is done by fitted.lm.

Examples

# GEL can deal with endogeneity problems

n = 200
phi <- c(.2, .7)
thet <- .2
sd <- .2
set.seed(123)
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = thet, sd = sd)), ncol = 1)
y <- x[7:n]
ym1 <- x[6:(n-1)]
ym2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y - ym1 + ym2
x <- H
res <- gel(g, x, c(0, .3, .6))
plot(y, main = "Fitted ARMA with GEL")
lines(fitted(res), col = 2)

# GMM is like GLS for linear models without endogeneity problems

set.seed(345)
n = 200
phi <- c(.2, .7)
thet <- 0
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = thet, sd = sd)), ncol = 1)
y <- 10 + 5*runorm(n) + x
res <- gmm(y ~ x, x)
plot(x, y, main = "Fitted model with GMM")
lines(x, fitted(res), col = 2)
legend("topleft", c("Y", "Yhat"), col = 1:2, lty = c(1,1))
**Description**

Method to extract the formula from \texttt{gel} or \texttt{gmm} objects.

**Usage**

```r
## S3 method for class 'gel'
formula(x, ...)
## S3 method for class 'gmm'
formula(x, ...)
```

**Arguments**

- \texttt{x} An object of class \texttt{gel} or \texttt{gmm} returned by the function \texttt{gel} or \texttt{gmm}
- \texttt{...} Other arguments to pass to other methods

**Examples**

```r
## GEL ##
n = 200
phi <- c(0.2, 0.7)
theta <- 0.3
sd <- .2
set.seed(123)
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
y <- x[7:n]
y1 <- x[6:(n-1)]
y2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y ~ y1 + y2
x <- H
res <- gel(g, x, c(0, 3.6))
formula(res)

# GMM is like GLS for linear models without endogeneity problems
set.seed(345)
n = 200
phi <- c(0.2, 0.7)
theta <- 0
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
y <- 10 + 5 * rnorm(n) + x
```
Description

Function to estimate a vector of parameters based on moment conditions using the GEL method as presented by Newey-Smith(2004) and Anatolyev(2005).

Usage

gel(g, x, tet0 = NULL, gradv = NULL, smooth = FALSE,
    type = c("EL", "ET", "CUE", "ETEL", "HD", "ETHD"),
    kernel = c("Truncated", "Bartlett"), bw = bwAndrews,
    approx = c("AR(1)", "ARMA(1,1)"), prewhite = 1, ar.method = "ols",
    tol_weights = 1e-7, tol_lam = 1e-9, tol_obj = 1e-9, tol_mom = 1e-9,
    maxiterlam = 100, constraint = FALSE, optfct = c("optim", "optimize",
                   "nlminb"), optlam = c("nlminb", "optim", "iter", "Wu"), data,
    Lambdacontrol = list(), model = TRUE, X = FALSE, Y = FALSE,
    TypeGel = "baseGel", alpha = NULL, eqConst = NULL,
    eqConstFullVcov = FALSE, onlyCoefficients=FALSE, ...)
evalGel(g, x, tet0, gradv = NULL, smooth = FALSE,
    type = c("EL", "ET", "CUE", "ETEL", "HD", "ETHD"),
    kernel = c("Truncated", "Bartlett"), bw = bwAndrews,
    approx = c("AR(1)", "ARMA(1,1)"), prewhite = 1,
    ar.method = "ols", tol_weights = 1e-7, tol_lam = 1e-9, tol_obj = 1e-9,
    tol_mom = 1e-9, maxiterlam = 100, optlam = c("nlminb", "optim",
                   "iter", "Wu"), data, Lambdacontrol = list(), model = TRUE, X = FALSE,
    Y = FALSE, alpha = NULL, ...)

Arguments

g    A function of the form \( g(\theta, x) \) and which returns a \( n \times q \) matrix with typical element \( g_i(\theta, x_t) \) for \( i = 1, \ldots, q \) and \( t = 1, \ldots, n \). This matrix is then used to build the q sample moment conditions. It can also be a formula if the model is linear (see details below).
tet0    A \( k \times 1 \) vector of starting values. If the dimension of \( \theta \) is one, see the argument "optfct". In the linear case, if tet0=NULL, the 2-step gmm estimator is used as starting value. However, it has to be provided when eqConst is not NULL
x    The matrix or vector of data from which the function \( g(\theta, x) \) is computed. If "g" is a formula, it is an \( n \times Nh \) matrix of instruments (see details below).
A function of the form $G(\theta, x)$ which returns a $q \times k$ matrix of derivatives of $\bar{g}(\theta)$ with respect to $\theta$. By default, the numerical algorithm numericDeriv is used. It is of course strongly suggested to provide this function when it is possible. This gradient is used compute the asymptotic covariance matrix of $\hat{\theta}$. If "g" is a formula, the gradient is not required (see the details below).

If set to TRUE, the moment function is smoothed as proposed by Kitamura(1997)


type

time of kernel used to compute the covariance matrix of the vector of sample moment conditions (see kernHAC for more details) and to smooth the moment conditions if "smooth" is set to TRUE. Only two types of kernel are available. The truncated implies a Bartlett kernel for the HAC matrix and the Bartlett implies a Parzen kernel (see Smith 2004).

bw

The method to compute the bandwidth parameter. By default it is bwAndrews which is proposed by Andrews (1991). The alternative is bwNeweyWest of Newey-West(1994).

logical or integer. Should the estimating functions be prewhitened? If TRUE or greater than 0 a VAR model of order as.integer(prewhite) is fitted via ar with method "ols" and demean = FALSE.

ar.method

character. The method argument passed to ar for prewhitening.

approx

a character specifying the approximation method if the bandwidth has to be chosen by bwAndrews.

tol.weights

numeric. Weights that exceed tol are used for computing the covariance matrix, all other weights are treated as 0.

tol.lam

Tolerance for $\lambda$ between two iterations. The algorithm stops when $\|\lambda_i - \lambda_{i-1}\|$ reaches tol.lam (see getLamb)

maxiterlam

The algorithm to compute $\lambda$ stops if there is no convergence after "maxiterlam" iterations (see getLamb).

tol.obj

Tolerance for the gradient of the objective function to compute $\lambda$ (see getLamb).

optfct

Only when the dimension of $\theta$ is 1, you can choose between the algorithm optim or optimize. In that case, the former is unreliable. If optimize is chosen, "t0" must be $1 \times 2$ which represents the interval in which the algorithm seeks the solution. It is also possible to choose the nlminb algorithm. In that case, borns for the coefficients can be set by the options upper= and lower=.

constraint

If set to TRUE, the constraint optimization algorithm is used. See constrOptim to learn how it works. In particular, if you choose to use it, you need to provide "ui" and "ci" in order to impose the constraint $ui \theta - ci \geq 0$.

tol.mom

It is the tolerance for the moment condition $\sum_{i=1}^{n} p_i g(\theta(x_i)) = 0$, where $p_i = \frac{1}{n} D \rho(< g_i, \lambda >)$ is the implied probability. It adds a penalty if the solution diverges from its goal.

optlam

Algorithm used to solve for the lagrange multiplier in getLamb. The algorithm Wu is only for type="EL".
data A data.frame or a matrix with column names (Optional).
Lambdacontrol Controls for the optimization of the vector of Lagrange multipliers used by either
optim, nlminb or constrOptim
model, X, Y logicals. If TRUE the corresponding components of the fit (the model frame, the
model matrix, the response) are returned if g is a formula.
TypeGel The name of the class object created by the method getModel. It allows developers to extend the package and create other GEL methods.
alpha Regularization coefficient for discrete CGEL estimation (experimental). By setting alpha to any value, the model is estimated by CGEL of type specified by the option type. See Chausse (2011)
eq const Either a named vector (if "g" is a function), a simple vector for the nonlinear
case indicating which of the \( \theta_0 \) is restricted, or a qx2 vector defining equality
constraints of the form \( \theta_i = c_i \). See gmm for an example.
eq constFullVcov If FALSE, the constrained coefficients are assumed to be fixed and only the co-
variance of the unconstrained coefficients is computed. If TRUE, the covariance
matrix of the full set of coefficients is computed.
onlyCoefficients If TRUE, only the vector of coefficients and Lagrange multipliers are returned
... More options to give to optim, optimize or constrOptim.

Details
If we want to estimate a model like \( Y_t = \theta_1 + X_{2t} \theta_2 + \ldots + X_{kt} \theta_k + \epsilon_t \) using the moment conditions
\( \text{Cov}(\epsilon_t H_t) = 0 \), where \( H_t \) is a vector of \( Nh \) instruments, then we can define "g" like we do for
lm. We would have \( g = y-x2+x3+\ldots+xk \) and the argument "x" above would become the matrix
\( H \) of instruments. As for \( \text{lm} \), \( Y_t \) can be a \( Ny \times 1 \) vector which would imply that \( k = Nh \times Ny \).
The intercept is included by default so you do not have to add a column of ones to the matrix \( H \).
You do not need to provide the gradient in that case since in that case it is embedded in gel. The
intercept can be removed by adding -1 to the formula. In that case, the column of ones need to be
added manually to \( H \).
If "smooth" is set to TRUE, the sample moment conditions \( \sum_{t=1}^{n} g(\theta, x_t) \) is replaced by: \( \sum_{t=1}^{n} g^k(\theta, x_t) \),
where \( g^k(\theta, x_t) = \sum_{i=-r}^{r} k(i) g(\theta, x_{t+i}) \), where \( r \) is a truncated parameter that depends on the
bandwidth and \( k(i) \) are normalized weights so that they sum to 1.
The method solves \( \hat{\theta} = \arg \min \left[ \arg \max_{\lambda} \frac{1}{n} \sum_{t=1}^{n} \rho(<g(\theta, x_t), \lambda>) - \rho(0) \right] \)
evalGel generates the object of class "gel" for a fixed vector of parameters. There is no estimation
for \( \theta \), but the optimal vector of Lagrange multipliers \( \lambda \) is computed. The objective function is then
the profiled likelihood for a given \( \theta \). It can be used to construct a confidence interval by inverting
the likelihood ratio test.

Value
'gel' returns an object of 'class' "gel"
The functions 'summary' is used to obtain and print a summary of the results.
The object of class "gel" is a list containing at least the following:
coefficients  \( k \times 1 \) vector of parameters
residuals  the residuals, that is response minus fitted values if "g" is a formula.
fitted.values  the fitted mean values if "g" is a formula.
lambda  \( q \times 1 \) vector of Lagrange multipliers.
vcov_par  the covariance matrix of "coefficients"
vcov_lambda  the covariance matrix of "lambda"
pt  The implied probabilities
objective  the value of the objective function
conv_lambda  Convergence code for "lambda" (see getLamb)
conv_mes  Convergence message for "lambda" (see getLamb)
conv_par  Convergence code for "coefficients" (see optim, optimize or constrOptim)
terms  the terms object used when g is a formula.
call  the matched call.
y  if requested, the response used (if "g" is a formula).
x  if requested, the model matrix used if "g" is a formula or the data if "g" is a function.
model  if requested (the default), the model frame used if "g" is a formula.

References


Examples

# First, an example with the function g()

```r
# x is a matrix

n <- nrow(x)
f <- cbind(u, u*x[4:(n-3)], u*x[3:(n-4)], u*x[2:(n-5)], u*x[1:(n-6)])
return(f)
```

```r
Dg <- function(tet,x) {
  n <- nrow(x)
  xx <- cbind(rep(1, (n-6)), x[6:(n-1)], x[5:(n-2)])
  H <- cbind(rep(1, (n-6)), x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
  f <- -crossprod(H, xx)/(n-6)
  return(f)
}
```

```r
n = 200
phi <- c(.2, .7)
theta <- .2
set.seed(123)
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
res <- gel(g, x, c(0, .3, .6), grad = Dg)
summary(res)
```

# The same model but with g as a formula.... much simpler in that case

```r
y <- x[7:n]
yml <- x[6:(n-1)]
ym2 <- x[5:(n-2)]

H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y ~ yml + ym2
x <- H
res <- gel(g, x, c(0, .3, .6))
summary(res)
```

# Using evalGel to create the object without estimation

```r
res <- evalGel(g, x, res$coefficients)
```
Description

It extract the data from a formula $y-z$ with instrument $h$ and put everything in a matrix. It helps redefine the function $g(\theta, x)$ that is required by \texttt{gmm} and \texttt{gel}.

Usage

\begin{verbatim}
getDat(formula, h, data, error=TRUE)
\end{verbatim}

Arguments

- \texttt{formula} A formula that defines the linear model to be estimated (see details).
- \texttt{h} A $n \times nh$ matrix of instruments (see details).
- \texttt{data} A data.frame or a matrix with colnames (Optionnal).
- \texttt{error} If FALSE, the data is generated without giving any error message

Details

The model to be estimated is based on the moment conditions $<h, (y-z\theta)> = 0$. It adds a column of ones to $z$ and $h$ by default. They are removed if -1 is added to the formula. The error argument has been added for \texttt{sysGmm} with common coefficients because the check is only valid for equation by equation identification.

Value

- $x$: A $n \times l$ matrix, where $l = ncol(y) + ncol(z) + ncol(h) + 2$ if "intercept" is TRUE and $ncol(y) + ncol(z) + xcol(h)$ if "intercept" is FALSE.
- $nh$: dimension of $h$
- $k$: dimension of $z$
- $ny$: dimension of $y$

Examples

\begin{verbatim}
n = 500
phi <- c(.2, .7)
theta <- 0.2
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
y <- x[7:n]
y1 <- x[6:(n-1)]
y2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
\end{verbatim}
getLamb

Solving for the Lagrange multipliers of Generalized Empirical Likelihood (GEL)

Description

It computes the vector of Lagrange multipliers, which maximizes the GEL objective function, using an iterative Newton method.

Usage

getLamb(gt, l0, type = c("EL","ET","CUE", "ETEL", "HD","ETHD"),
        tol_lam = 1e-7, maxiterlam = 100,
        tol_obj = 1e-7, k = 1, method = c("nlminb", "optim", "iter", "Wu"),
        control = list())

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gt</td>
<td>A $n \times q$ matrix with typical element $g_i(\theta, x_i)$</td>
</tr>
<tr>
<td>l0</td>
<td>Vector of starting values for lambda</td>
</tr>
<tr>
<td>type</td>
<td>&quot;EL&quot; for empirical likelihood, &quot;ET&quot; for exponential tilting, &quot;CUE&quot; for continuous updated estimator, and &quot;HD&quot; for Hellinger Distance. See details for &quot;ETEL&quot; and &quot;ETHD&quot;.</td>
</tr>
<tr>
<td>tol_lam</td>
<td>Tolerance for $\lambda$ between two iterations. The algorithm stops when $|\lambda_i - \lambda_{i-1}|$ reaches tol_lam</td>
</tr>
<tr>
<td>maxiterlam</td>
<td>The algorithm stops if there is no convergence after &quot;maxiterlam&quot; iterations.</td>
</tr>
<tr>
<td>tol_obj</td>
<td>Tolerance for the gradient of the objective function. The algorithm returns a non-convergence message if $\max(</td>
</tr>
<tr>
<td>k</td>
<td>It represents the ratio $k_1/k_2$, where $k_1 = \int_{-\infty}^{\infty} k(s)ds$ and $k_2 = \int_{-\infty}^{\infty} k(s)^2 ds$. See Smith(2004).</td>
</tr>
<tr>
<td>method</td>
<td>The iterative procedure uses a Newton method for solving the FOC. It however recommended to use optim or nlminb. If type is set to &quot;EL&quot; and method to &quot;optim&quot;, constrOptim is called to prevent $\log(1 - gt^T \lambda)$ from producing NA. The gradient and hessian is provided to nlminb which speed up the convergence. The latter is therefore the default value. &quot;Wu&quot; is for &quot;EL&quot; only. It uses the algorithm of Wu (2005).</td>
</tr>
<tr>
<td>control</td>
<td>Controls to send to optim, nlminb or constrOptim</td>
</tr>
</tbody>
</table>

```r
x <- getDat(y - ym1 + ym2, H)
```
getModel

Details

It solves the problem \[ \max_{\lambda} \frac{1}{n} \sum_{t=1}^{n} \rho(g^t \lambda). \] For the type "ETEL", it is only used by `gel`. In that case \( \lambda \) is obtained by maximizing \( \frac{1}{n} \sum_{t=1}^{n} \rho(g^t \lambda) \), using \( \rho(v) = -\exp v \) (so ET) and \( \theta \) by minimizing the same equation but with \( \rho(v) = -\log(1-v) \). To avoid NA's, `constrOptim` is used with the restriction \( X g_t < 1 \). The type "ETHD" is experimental and proposed by Antoine-Dovonon (2015). The paper is not yet available.

Value

lambda: A \( q \times 1 \) vector of Lagrange multipliers which solve the system of equations given above.
conv: Details on the type of convergence.

References


Examples

```r
g <- function(tet, x)
{
  n <- nrow(x)
  f <- cbind(u, u*x[4:(n-3)], u*x[3:(n-4)], u*x[2:(n-5)], u*x[1:(n-6)])
  return(f)
}

n = 500
tet <- c(.2, .7)
phi <- c(.2, .7)
theta <- 0.2
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
gt <- g(c(0, phi), x)
getLamb(gt, type = "EL", method="optim")
```

getModel

Method for setting the properties of a model

Description

It collects what is needed by the method `momentEstim` (see details).
Usage

```r
## S3 method for class 'baseGmm'
getModel(object, ...)
## S3 method for class 'sysGmm'
getModel(object, ...)
## S3 method for class 'baseGel'
getModel(object, ...)
## S3 method for class 'constGel'
getModel(object, ...)
## S3 method for class 'constGel'
getModel(object, ...)
## S3 method for class 'tsls'
getModel(object, ...)
## S3 method for class 'ateGel'
getModel(object, ...)
```

Arguments

- `object`: An object of class `baseGmm`
- `...`: Other arguments when `getModel` is applied to another class object

Value

It returns an object of the right class which determines how the method `momentEstim` will treat it. For example, if `g` is a formula and `type` is set to "cue", it creates an object of class `baseGmm` with formula. It this case, `momentEstim`, applied to this object, computes the continuously updated GMM of a linear model. It allows more flexibility this way. For example, it could be easy to add a GMM method which is robust in presence of weak identification simply by creating a new class of model and the associated `momentEstim` method.

---

**gmm**

*Generalized method of moment estimation*

Description

Function to estimate a vector of parameters based on moment conditions using the GMM method of Hansen(82).

Usage

```r
gmm(g,x,t0=NULL,gradv=NULL, type=c("twoStep","cue","iterative"),
    wmatrix = c("optimal","ident"), vcov=c("HAC","MDS","iid","TrueFixed"),
    kernel=c("Quadratic Spectral","Truncated","Bartlett","Parzen","Tukey-Hanning"),
    crit=1e-7,bw = bwAndrews, prewhite = 1, ar.method = "ols", approx="AR(1)",
    tol = 1e-7, itermax=100, optfct=c("optim","optimize","nlminb", "constrOptim"),
    model=TRUE, X=FALSE, Y=FALSE, TypeGmm = "baseGmm", centeredVcov = TRUE,
```
weightsMatrix = NULL, traceIter = FALSE, data, eqConst = NULL, eqConstFullVcov = FALSE, mustar = NULL, onlyCoefficients=FALSE, ...)
evalGmm(g, x, t0, tetw=NULL, gradv=NULL, wmatrix = c("optimal","ident"), vcov=c("HAC","iid","TrueFixed"), kernel=c("Quadratic Spectral","Truncated", "Bartlett", "Parzen", "Tukey-Hanning"), crit=1e-7, bw = bwAndrews, prewhite = FALSE, ar.method = "ols", approx="AR(1)", tol = 1e-7, model=TRUE, X=FALSE, Y=FALSE, centeredVcov = TRUE, weightsMatrix = NULL, data, mustar = NULL)
gmmWithConst(obj, which, value)

Arguments

**g**
A function of the form $g(\theta, x)$ and which returns a $n \times q$ matrix with typical element $g_i(\theta, x_t)$ for $i = 1, \ldots, q$ and $t = 1, \ldots, n$. This matrix is then used to build the q sample moment conditions. It can also be a formula if the model is linear (see details below).

**x**
The matrix or vector of data from which the function $g(\theta, x)$ is computed. If "g" is a formula, it is an $n \times Nh$ matrix of instruments or a formula (see details below).

**t0**
A $k \times 1$ vector of starting values. It is required only when "g" is a function because only then a numerical algorithm is used to minimize the objective function. If the dimension of $\theta$ is one, see the argument "optfct".

**tetw**
A $k \times 1$ vector to compute the weighting matrix.

**gradv**
A function of the form $G(\theta, x)$ which returns a $q \times k$ matrix of derivatives of $\bar{g}(\theta)$ with respect to $\theta$. By default, the numerical algorithm numericDeriv is used. It is of course strongly suggested to provide this function when it is possible. This gradient is used to compute the asymptotic covariance matrix of $\theta$ and to obtain the analytical gradient of the objective function if the method is set to "CG" or "BFGS" in optim and if "type" is not set to "cue". If "g" is a formula, the gradient is not required (see the details below).

**type**
The GMM method: "twostep" is the two step GMM proposed by Hansen(1982) and the "cue" and "iterative" are respectively the continuous updated and the iterative GMM proposed by Hansen, Eaton et Yaron (1996)

**wmatrix**
Which weighting matrix should be used in the objective function. By default, it is the inverse of the covariance matrix of $g(\theta, x)$. The other choice is the identity matrix which is usually used to obtain a first step estimate of $\theta$

**vcov**
Assumption on the properties of the random vector x. By default, x is a weakly dependant process. The "iid" option will avoid using the HAC matrix which will accelerate the estimation if one is ready to make that assumption. The option "TrueFixed" is used only when the matrix of weights is provided and it is the optimal one.

**kernel**
type of kernel used to compute the covariance matrix of the vector of sample moment conditions (see kernHAC for more details)

**crit**
The stopping rule for the iterative GMM. It can be reduce to increase the precision.
bw
The method to compute the bandwidth parameter in the HAC weighting matrix. The default is `link{bwAndrews}` (as proposed in Andrews (1991)), which minimizes the MSE of the weighting matrix. Alternatives are `link{bwWilhelm}` (as proposed in Wilhelm (2015)), which minimizes the mean-square error (MSE) of the resulting GMM estimator, and `link{bwNeweyWest}` (as proposed in Newey-West(1994)).

prewhite
logical or integer. Should the estimating functions be prewhitened? If `TRUE` or greater than 0 a VAR model of order `as.integer(prewhite)` is fitted via `ar` with method "ols" and `demean = FALSE`.

ar.method
character. The method argument passed to `ar` for prewhitening.

approx
A character specifying the approximation method if the bandwidth has to be chosen by `bwAndrews`.

tol
Weights that exceed `tol` are used for computing the covariance matrix, all other weights are treated as 0.

itermax
The maximum number of iterations for the iterative GMM. It is unlikely that the algorithm does not converge but we keep it as a safety.

optfct
Only when the dimension of \( \theta \) is 1, you can choose between the algorithm `optim` or `optimize`. In that case, the former is unreliable. If `optimize` is chosen, "t0" must be \( 1 \times 2 \) which represents the interval in which the algorithm seeks the solution. It is also possible to choose the `nlminb` algorithm. In that case, boundaries for the coefficients can be set by the options `upper=` and `lower=`. The `constrOptim` is only available for nonlinear models for now. The standard errors may have to be corrected if the estimates reach the boundary set by `ui` and `ci`.

model, X, Y
logical. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response) are returned if `g` is a formula.

TypeGmm
The name of the class object created by the method `getModel`. It allows developers to extend the package and create other GMM methods.

centeredVcov
Should the moment function be centered when computing its covariance matrix. Doing so may improve inference.

weightsMatrix
It allows users to provide `gmm` with a fixed weighting matrix. This matrix must be \( q \times q \), symmetric and strictly positive definite. When provided, the `type` option becomes irrelevant.

traceIter
Tracing information for GMM of type "iter"

data
A data.frame or a matrix with column names (Optional).

eqConst
Either a named vector (if "g" is a function), a simple vector for the nonlinear case indicating which of the \( \theta_0 \) is restricted, or a qx2 vector defining equality constraints of the form \( \theta_i = c_i \). See below for an example.

which, value
The equality constraint is of the form `which=value`. "which" can be a vector of type characters with the names of the coefficients being constrained, or a vector of type numeric with the position of the coefficient in the whole vector.

obj
Object of class "gmm"
eqConstFullVcov

If FALSE, the constrained coefficients are assumed to be fixed and only the covariance of the unconstrained coefficients is computed. If TRUE, the covariance matrix of the full set of coefficients is computed.

mustar

If not null, it must be a vector with the number of elements being equal to the number of moment conditions. In that case, the vector is subtracted from the sample moment vector before minimizing the objective function. It is useful to do a bootstrap procedure.

onlyCoefficients

If set to TRUE, the function only returns the coefficient estimates. It may be of interest when the standard errors are not needed.

Details

If we want to estimate a model like $Y_t = \theta_1 + X_2\theta_2 + \cdots + X_k\theta_k + \epsilon_t$ using the moment conditions $\text{Cov}(\epsilon_t, H_t) = 0$, where $H_t$ is a vector of $Nh$ instruments, then we can define "g" like we do for lm. We would have $g = y\tilde{x}2 + x3 + \cdots + xk$ and the argument "x" above would become the matrix H of instruments. As for lm, $Y_t$ can be a $Ny \times 1$ vector which would imply that $k = Nh \times Ny$. The intercept is included by default so you do not have to add a column of ones to the matrix $H$. You do not need to provide the gradient in that case since in that case it is embedded in gmm. The intercept can be removed by adding -1 to the formula. In that case, the column of ones need to be added manually to $H$. It is also possible to express "x" as a formula. For example, if the instruments are $\{1, z1, z2, z3\}$, we can set "x" to $z1 + z2 + z3$. By default, a column of ones is added. To remove it, set "x" to $z1 + z2 + z3 - 1$.

The following explains the last example bellow. Thanks to Dieter Rozenich, a student from the Vienna University of Economics and Business Administration. He suggested that it would help to understand the implementation of the Jacobian.

For the two parameters of a normal distribution $(\mu, \sigma)$ we have the following three moment conditions:

$m_1 = \mu - x_i$

$m_2 = \sigma^2 - (x_i - \mu)^2$

$m_3 = x_i^3 - \mu(\mu^2 + 3\sigma^2)$

$m_1, m_2$ can be directly obtained by the definition of $(\mu, \sigma)$. The third moment condition comes from the third derivative of the moment generating function (MGF)

$$M_X(t) = \exp(\mu t + \frac{\sigma^2 t^2}{2})$$

evaluated at $(t = 0)$.

Note that we have more equations (3) than unknown parameters (2).

The Jacobian of these two conditions is (it should be an array but I can’t make it work):

$$
\begin{bmatrix}
1 & 0 \\
-2\mu + 2x & 2\sigma
\end{bmatrix}
$$
\[-3\mu^2 - 3\sigma^2 - 6\mu\sigma\]

gmmWithConst() re-estimates an unrestricted model by adding an equality constraint. evalGmm() creates an object of class "gmm" for a given parameter vector. If no vector "tew" is provided and the weighting matrix needs to be computed, "t0" is used.

**Value**

'gmm' returns an object of 'class' "gmm"

The functions 'summary' is used to obtain and print a summary of the results. It also compute the J-test of overidentifying restriction

The object of class "gmm" is a list containing at least:

- **coefficients**: \( k \times 1 \) vector of coefficients
- **residuals**: the residuals, that is response minus fitted values if "g" is a formula.
- **fitted.values**: the fitted mean values if "g" is a formula.
- **vcov**: the covariance matrix of the coefficients
- **objective**: the value of the objective function \( \| \text{var}(\bar{g})^{-1/2} \bar{g} \|^2 \)
- **terms**: the terms object used when g is a formula.
- **call**: the matched call.
- **y**: if requested, the response used (if "g" is a formula).
- **x**: if requested, the model matrix used if "g" is a formula or the data if "g" is a function.
- **model**: if requested (the default), the model frame used if "g" is a formula.
- **algoInfo**: Information produced by either optim or nlminb related to the convergence if "g" is a function. It is printed by the summary.gmm method.

**References**


Examples

```r
## CAPM test with GMM
data(Finance)
r <- Finance[1:300, 1:10]
rm <- Finance[1:300, "rm"]
rf <- Finance[1:300, "rf"]
z <- as.matrix(r-rf)
t <- nrow(z)
zm <- rm-rf
h <- matrix(zm, t, 1)
res <- gmm(z ~ zm, x = h)
summary(res)

## linear tests can be performed using linearHypothesis from the car package
## The CAPM can be tested as follows:
library(car)
linearHypothesis(res,cbind(diag(10),matrix(0,10,10)),rep(0,10))

# The CAPM of Black
g <- function(theta, x) {
gmat <- cbind(e, e*c(x[,1]))
return(gmat) }

x <- as.matrix(cbind(rm, r))
res_black <- gmm(g, x = x, t0 = rep(0, 11))

summary(res_black)$coefficients

## APT test with Fama-French factors and GMM
f1 <- zm
f2 <- Finance[1:300, "hml" ]
f3 <- Finance[1:300, "smb"]
h <- cbind(f1, f2, f3)
res2 <- gmm(z ~ f1 + f2 + f3, x = h)
coef(res2)
summary(res2)$coefficients

## Same result with x defined as a formula:
res2 <- gmm(z ~ f1 + f2 + f3, ~ f1 + f2 + f3)
coef(res2)

## The following example has been provided by Dieter Rozenich (see details).
# It generates normal random numbers and uses the GMM to estimate
# mean and sd.
#---------------------------------------------------------------
```
# Random numbers of a normal distribution
# First we generate normally distributed random numbers and compute the two parameters:
n <- 1000
x <- rnorm(n, mean = 4, sd = 2)
# Implementing the 3 moment conditions
g <- function(tet, x)
  {
    m1 <- (tet[1] - x)
m2 <- (tet[2]^2 - (x - tet[1])^2)
m3 <- x^3 - tet[1]*x(tet[1]^2 + 3*tet[2]^2)
f <- cbind(m1, m2, m3)
return(f)
  }
# Implementing the jacobian
Dg <- function(tet, x)
  {
    jacobian <- matrix(c(1, 2*x(-tet[1]+mean(x)), -3*tet[1]^2-3*tet[2]^2, 0, 2*tet[2],
      -6*tet[1]*tet[2]), nrow=3, ncol=2)
    return(jacobian)
  }
# Now we want to estimate the two parameters using the GMM.
gmm(g, x, c(0, 0), grad = Dg)
# Two-stage-least-squares (2SLS), or IV with iid errors.
# The model is:
# Y(t) = b[0] + b[1]C(t) + b[2]Y(t-1) + e(t)
# e(t) is an MA(1)
# The instruments are Z(t)=[1 C(t) Y(t-2) y(t-3) y(t-4))
getdat <- function(n) {
  e <- arima.sim(n, model=list(ma=.9))
  C <- runif(n,0,5)
  Y <- rep(0,n)
  for (i in 2:n){
    Y[i] = 1 + 2*C[i] + 0.9*Y[i-1] + e[i]
  }
  Yt <- Y[5:n]
  X <- cbind(1,C[5:n],Y[4:(n-1)])
  Z <- cbind(1,C[5:n],Y[3:(n-2)],Y[2:(n-3)],Y[1:(n-4)])
  return(list(Y=Yt,X=X,Z=Z))
}
d <- getdat(5000)
res4 <- gmm(d$Y,d$X-1,-d$Z-1,vcov="iid")
res4

### Examples with equality constraint

# Random numbers of a normal distribution

### Not run:
gmm

# The following works but produces warning message because the dimension of coef is 1
# Brent should be used

# without named vector
# Method Brent is used because the problem is now one-dimensional
gmm(g, x, c(4, 0), grad = Dg, eqConst=1, method="Brent", lower=-10, upper=10)

# with named vector
gmm(g, x, c(mu=4, sig=2), grad = Dg, eqConst="sig", method="Brent", lower=-10, upper=10)

## End(Not run)

gmm(g, x, c(4, 0), grad = Dg, eqConst=1, method="Brent", lower=0, upper=6)
gmm(g, x, c(mu=4, sig=2), grad = Dg, eqConst="sig", method="Brent", lower=0, upper=6)

# Example with formula
# first coef = 0 and second coef = 1
# Only available for one dimensional yt

z <- z[,1]
res2 <- gmm(z ~ f1 + f2 + f3, - f1 + f2 + f3, eqConst = matrix(c(1,2,0,1),2,2))
res2

# CUE with starting t0 requires eqConst to be a vector
res3 <- gmm(z ~ f1 + f2 + f3, - f1 + f2 + f3, t0=c(0,1,.5,.5), type="cue", eqConst = c(1,2))
res3

#### Examples with equality constraints, where the constrained coefficients is used to compute
#### the covariance matrix.
#### Useful when some coefficients have been estimated before, they are just identified in GMM
#### and don't need to be re-estimated.
#### To use with caution because the covariance won't be valid if the coefficients do not solve
#### the GMM FOC.

res4 <- gmm(z ~ f1 + f2 + f3, - f1 + f2 + f3, t0=c(0,1,.5,.5), eqConst = c(1,2), eqConstFullVcov=TRUE)
summary(res4)

#### Examples with equality constraint using gmmWithConst

res2 <- gmm(z ~ f1 + f2 + f3, - f1 + f2 + f3)
gmmWithConst(res2,c("f2","f3"),c(.5,.5))
gmmWithConst(res2,c(2,3),c(.5,.5))

## Creating an object without estimation for a fixed parameter vector

res2_2 <- evalGmm(z ~ f1 + f2 + f3, - f1 + f2 + f3, t0=dat$coefficients, tetw=dat$coefficients)
summary(res2_2)
Growth Data

Description

Panel of Macroeconomic data for 125 countries from 1960 to 1985 constructed by Summers and Heston (1991))

Usage

data(Growth)

Format

A data frame containing 9 vectors.

- **Country_ID**: Country identification number
- **COM**: 1 if the country is in a communist regime, 0 otherwise
- **OPEC**: 1 if the country is part of the OPEC, 0 otherwise
- **Year**: Year
- **GDP**: Per capita GDP (in thousands) in 1985 U.S. dollars.
- **LagGDP**: GDP of the previous period
- **SavRate**: Saving rate measured as the ratio of real investment to real GDP
- **LagSavRate**: SavRate of the previous period
- **Country**: Country names
- **Pop**: Population in thousands
- **LagPop**: Population of the previous period

Source

http://fhayashi.fc2web.com/datasets.htm
Description

The test is proposed by Kleibergen (2005). It is robust to weak identification.

Usage

\[
\text{KTest}(\text{obj}, \theta_0 = \text{NULL}, \alpha_K = 0.04, \alpha_J = 0.01)
\]

## S3 method for class 'gmmTests'

\[
\text{print}(x, \text{digits} = 5, \ldots)
\]

Arguments

- **obj**: Object of class "gmm" returned by \texttt{gmm}
- **theta0**: The null hypothesis being tested. See details.
- **alphaK, alphaJ**: The size of the J and K tests when combining the two. The overall size is \(\alpha_K + \alpha_J\).
- **x**: An object of class \texttt{gmmTests} returned by \texttt{KTest}
- **digits**: The number of digits to be printed
- **\ldots**: Other arguments when \texttt{print} is applied to another class object

Details

The function produces the J-test and K-statistics which are robust to weak identification. The test is either \(H_0 : \theta = \theta_0\), in which case \(\theta_0\) must be provided, or \(\beta = \beta_0\), where \(\theta = (\alpha', \beta')'\), and \(\alpha\) is assumed to be identified. In the latter case, \(\theta_0\) is \text{NULL} and \text{obj} is a restricted estimation in which \(\beta\) is fixed to \(\beta_0\). See \texttt{gmm} and the option "eqConst" for more details.

Value

Tests and p-values

References

Keibergen, F. (2005), Testing Parameters in GMM without assuming that they are identified. \textit{Econometrica}, \textbf{73}, 1103-1123;

Examples

```
library(mvtnorm)
sig <- matrix(c(1,.5,.5,1),2,2)
n <- 400
e <- rmvnorm(n,sigma=sig)
x4 <- rnorm(n)
```
w <- exp(-x^2) + e[,1]
y <- 0.1*w + e[,2]
h <- cbind(x4, x4^2, x4^3, x4^6)
g3 <- y-w
res <- gmm(g3,h)

# Testing the whole vector:
KTest(res,theta0=c(0,.1))

# Testing a subset of the vector (See \code{\link{gmm}})
res2 <- gmm(g3, h, eqConst=matrix(c(2,1),1,2))
res2
KTest(res2)

marginal

\textit{Marginal effects Summary}

\section*{Description}

It produces the summary table of marginal effects for GLM estimation with GEL. Only implemented for ATEgel.

\section*{Usage}

\texttt{## S3 method for class \textquoteleft ategel	extquoteright}
marginal(object, ...)

\section*{Arguments}

\begin{itemize}
  \item \texttt{object} \hspace{1cm} An object of class ategel returned by the function \texttt{ATEgel}
  \item \texttt{...} \hspace{1cm} Other arguments for other methods
\end{itemize}

\section*{Value}

It returns a matrix with the marginal effects, the standard errors based on the Delta method when the link is nonlinear, the t-ratios, and the pvalues.

\section*{References}

Examples

```r
# We create some artificial data with unbalanced groups and binary outcome
genDat <- function(n)
{
  eta=c(-1, .5, -.25, -.1)
  Z <- matrix(rnorm(n*4), ncol=4)
  b <- c(27.4, 13.7, 13.7, 13.7)
  bZ <- c(Z%*%b)
  Y1 <- as.numeric(rnorm(n, mean=210+bZ)>220)
  Y0 <- as.numeric(rnorm(n, mean=200-.5*bZ)>220)
  etaZ <- c(Z%*%eta)
  pZ <- exp(etaZ)/(1+exp(etaZ))
  T <- rbinom(n, 1, pZ)
  Y <- T*Y1+(1-T)*Y0
  X1 <- exp(Z[,1]/2)
  X2 <- Z[,2]/(1+exp(Z[,1]))
  X3 <- (Z[,1]+Z[,3]/25+0.6)^3
  X4 <- (Z[,2]+Z[,4]+20)^2
  data.frame(Y=Y, cbind(X1,X2,X3,X4), T=T)
}

dat <- genDat(200)
res <- ategel(Y~T, ~X1+X2+X3+X4, data=dat, type="ET", family="logit")
summary(res)
marginal(res)
```

momentEstim

Method for estimating models based on moment conditions

Description

It estimates a model which is characterized by the method `getModel` (see details).

Usage

```r
## S3 method for class 'baseGmm.twostep'
momentEstim(object, ...)
## S3 method for class 'baseGmm.twostep.formula'
momentEstim(object, ...)
## S3 method for class 'sysGmm.twostep.formula'
momentEstim(object, ...)
## S3 method for class 'tsls.twostep.formula'
momentEstim(object, ...)
```
## Arguments

- **object**: An object created by the method `get_model`
- **...**: Other arguments when `moment_estim` is applied to another class object

## Value

It returns an object of class determined by the argument "TypeGMM" of `gmm`. By default, it is of class `baseGmm.res`. It estimates the model and organizes the results that will be finalized by the method `fin_res`. More methods can be created in order to use other GMM methods not yet included in the package.

## References


### nsw

**Lalonde subsample of the National Supported Work Demonstration Data (NSW)**

## Description

This data was collected to evaluate the National Supported Work (NSW) Demonstration project in Lalonde (1986).

## Usage

```r
data(nsw)
```
Format

A data frame containing 9 variables.

- **treat**: Treatment assignment
- **age**: Age
- **ed**: Years of Education
- **black**: 1 if Black, 0 otherwise
- **hisp**: 1 if Hispanic 0 otherwise
- **married**: 1 if married 0 otherwise
- **nodeg**: 1 if no college degree 0 otherwise
- **re75**: 1975 earnings
- **re78**: 1978 earnings

Details

The dataset was obtained from the ATE package (see reference).

Source

"NSW Data Files" from Rajeev Dehejia's website. URL: http://users.nber.org/~rdehejia/data/nswdata2.html


References


Usage

## S3 method for class 'gel'
plot(x, which = c(1:4),
    main = list("Residuals vs Fitted values", "Normal Q-Q",
                "Response variable and fitted values","Implied probabilities"),
    panel = if(add.smooth) panel.smooth else points,
    ask = prod(par("mfcol")) < length(which) && dev.interactive(), ..., 
    add.smooth = getOption("add.smooth"))

## S3 method for class 'gmm'
plot(x, which = c(1:3),
    main = list("Residuals vs Fitted values", "Normal Q-Q",
                "Response variable and fitted values"),
    panel = if(add.smooth) panel.smooth else points,
    ask = prod(par("mfcol")) < length(which) && dev.interactive(), ..., 
    add.smooth = getOption("add.smooth"))

Arguments

- **x** gel or gmm object, typically result of gel or gmm.
- **which** if a subset of the plots is required, specify a subset of the numbers 1:4 for gel or 1:3 for gmm.
- **main** Vector of titles for each plot.
- **panel** panel function. The useful alternative to points, panel.smooth can be chosen by add.smooth = TRUE.
- **ask** logical; if TRUE, the user is asked before each plot, see par(ask=.).
- **...** other parameters to be passed through to plotting functions.
- **add.smooth** logical indicating if a smoother should be added to most plots; see also panel above.

Details

It is a beta version of a plot method for gel objects. It is a modified version of plot.lm. For now, it is available only for linear models expressed as a formula. Any suggestions are welcome regarding plots or options to include. The first two plots are the same as the ones provided by plot.lm, the third is the dependant variable \( y \) with its mean \( \hat{y} \) (the fitted values) and the last plots the implied probabilities with the empirical density \( 1/T \).

Examples

```r
# GEL #
n = 500
phi <- c(0.2, 0.7)
theta <- 0
sd <- 0.2
```
print

\[ x \leftarrow \text{matrix}(\text{arima.sim}(n = n, \text{list}(\text{order} = c(2, 0, 1), \text{ar} = \phi, \text{ma} = \theta, \text{sd} = \text{sd})), \text{ncol} = 1) \]
\[ y \leftarrow x[7:n] \]
\[ ym1 \leftarrow x[6:(n-1)] \]
\[ ym2 \leftarrow x[5:(n-2)] \]
\[ H \leftarrow \text{cbind}(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)]) \]
\[ g \leftarrow y - ym1 + ym2 \]
\[ x \leftarrow H \]
\[ t0 \leftarrow c(0, 5, 5) \]
\[ \text{res} \leftarrow \text{gel}(g, x, t0) \]
\[ \text{plot(res, which} = 3) \]
\[ \text{plot(res, which} = 4) \]

# GMM #
\[ \text{res} \leftarrow \text{gmm}(g, x) \]
\[ \text{plot(res, which} = 3) \]

---

print \hspace{1cm} Printing a gmm or gel object

Description

It is a printing method for gmm or gel objects.

Usage

```r
## S3 method for class 'gmm'
\text{print}(x, \text{digits} = 5, ...) \\
## S3 method for class 'gel'
\text{print}(x, \text{digits} = 5, ...) \\
## S3 method for class 'sysGmm'
\text{print}(x, \text{digits} = 5, ...) 
```

Arguments

- `x`: An object of class gmm or gel returned by the function gmm or gel
- `digits`: The number of digits to be printed
- `...`: Other arguments when print is applied to an other class object

Value

It prints some results from the estimation like the coefficients and the value of the objective function.
Examples

```r
# GMM #

n = 500
phi <- c(0.2, 0.7)
theta <- 0
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd), ncol = 1))
y <- x[7:n]
yml1 <- x[6:(n-1)]
yml2 <- x[5:(n-2)]

H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y - yml1 + yml2
x <- H

res <- gmm(g, x)
print(res)

# GEL #

t0 <- c(0, 5, 5)
res <- gel(g, x, t0)
print(res)
```

residuals

Residuals of GEL or GMM

Description

Method to extract the residuals of the model estimated by gmm or gel.

Usage

```r
## S3 method for class 'gel'
residuals(object, ...)  
## S3 method for class 'gmm'
residuals(object, ...)
```

Arguments

- `object` An object of class gmm or gel returned by the function `gmm` or `gel`
- `...` Other arguments when `residuals` is applied to an other classe object

Value

It returns the matrix of residuals \((y - \hat{y})\) in \(g = y \sim x\) as it is done by `residuals.lm`.  

residuals
smoothG

Examples

# GEL can deal with endogeneity problems

n <- 200
phi <- c(0.2, 0.7)
theta <- 0.2
sd <- 0.2
set.seed(123)
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
y <- x[7:n]
y1 <- x[6:(n-1)]
y2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y ~ y1 + y2
x <- H
res <- gel(g, x, c(0.3, 0.6))
e <- residuals(res)
plot(e, type = "l", main = "Residuals from an ARMA fit using GEL")

# GMM is like GLS for linear models without endogeneity problems

set.seed(345)
n <- 200
phi <- c(0.2, 0.7)
theta <- 0
sd <- 0.2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
y <- 10 + 5*runif(n) + x
res <- gmm(y ~ x, x)
plot(x, residuals(res), main = "Residuals of an estimated model with GMM")

smoothG

Kernel smoothing of a matrix of time series

Description

It applies the required kernel smoothing to the moment function in order for the GEL estimator to be valid. It is used by the gel function.

Usage

smoothG(x, bw = bwAndrews, prewhite = 1, ar.method = "ols", weights = weightsAndrews,
kernel = c("Bartlett", "Parzen", "Truncated", "Tukey-Hanning"),
approx = c("AR(1)", "ARMA(1,1)"), tol = 1e-7)
Arguments

\(x\) a \(n \times q\) matrix of time series, where \(n\) is the sample size.

\(bw\) The method to compute the bandwidth parameter. By default, it uses the bandwidth proposed by Andrews (1991). As an alternative, we can choose \(bw=bw\text{NeweyWest}\) (without "") which is proposed by Newey-West (1996).

prewhite logical or integer. Should the estimating functions be prewhitened? If TRUE or greater than 0 a VAR model of order as.integer(prewhite) is fitted via ar with method "ols" and demean = FALSE.

ar.method character. The method argument passed to ar for prewhitening.

weights The smoothing weights can be computed by weightsAndrews of it can be provided manually. If provided, it has to be a \(r \times 1\) vector (see details).

approx a character specifying the approximation method if the bandwidth has to be chosen by bwAndrews.

tol numeric. Weights that exceed tol are used for computing the covariance matrix, all other weights are treated as 0.

kernel The choice of kernel

Details

The sample moment conditions \(\sum_{t=1}^{n} g(\theta, x_t)\) is replaced by: \(\sum_{t=1}^{n} g^k(\theta, x_t) = \sum_{i=-r}^{r} k(i) g(\theta, x_{t+i})\), where \(r\) is a truncated parameter that depends on the bandwidth and \(k(i)\) are normalized weights so that they sum to 1.

If the vector of weights is provided, it gives only one side weights. For example, if you provide the vector \((1,.5,.25)\), \(k(i)\) will become \((.25,.5,1,.5,.25)/(.25+.5+1+.5+.25) = (.1,.2,.4,.2,.1)\)

Value

smoothx: A \(q \times q\) matrix containing an estimator of the asymptotic variance of \(\sqrt{n} \bar{x}\), where \(\bar{x}\) is \(q \times 1\) vector with typical element \(\bar{x}_i = \frac{1}{n} \sum_{j=1}^{n} x_{ji}\). This function is called by gel but can also be used by itself.

kern_weights: Vector of weights used for the smoothing.

References


Examples

g <- function(tet, x)
{
  n <- nrow(x)
  f <- cbind(u, u*x[4:(n-3)], u*x[3:(n-4)], u*x[2:(n-5)], u*x[1:(n-6)])
  return(f)
}

n = 500
phi <- c(.2, .7)
theta <- 0.2
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2, 0, 1), ar = phi, ma = theta, sd = sd)), ncol = 1)
gt <- g(c(0, phi), x)
sgt <- smoothG(gt)$smoothx
plot(gt[,1])
lines(sgt[,1])

specTest

Description

Generic function for testing the specification of estimated models. It computes the J-test from \texttt{gmm}
objects and J-test, LR-test and LM-test from \texttt{gel} objects.

Usage

\#
\#
\#
## S3 method for class 'gmm'
specTest(x, ...)
## S3 method for class 'gel'
specTest(x, ...)
## S3 method for class 'specTest'
print(x, digits = 5, ...)
specTest(x, ...)

Arguments

x \hspace{1cm} A fitted model object.
digits \hspace{1cm} The number of digits to be printed.
... \hspace{1cm} Arguments passed to methods.

Value

Tests and p-values
References


Smith, R. J. (2004), GEL Criteria for Moment Condition Models. *CeMMAP working papers, Institute for Fiscal Studies*

Examples

```r
# Summary Method for object of class gmm or gel

summary

## Method for object of class gmm or gel

### Description

It presents the results from the gmm or gel estimation in the same fashion as summary does for the lm class objects for example. It also compute the tests for overidentifying restrictions.

### Usage

```r
## S3 method for class 'gmm'
summary(object, ...)  
## S3 method for class 'sysGmm'
summary(object, ...)  
## S3 method for class 'gel'
summary(object, ...)
```
summary

## S3 method for class 'ategel'
summary(object, robToMiss = TRUE, ...)

## S3 method for class 'tsls'
summary(object, vcov = NULL, ...)

## S3 method for class 'summary.gmm'
print(x, digits = 5, ...)

## S3 method for class 'summary.sysGmm'
print(x, digits = 5, ...)

## S3 method for class 'summary.gel'
print(x, digits = 5, ...)

## S3 method for class 'summary.tls'
print(x, digits = 5, ...)

Arguments

- **object** An object of class gmm or gel returned by the function gmm or gel
- **x** An object of class summary.gmm or summary.gel returned by the function summary.gmm summary.gel
- **digits** The number of digits to be printed
- **vcov** An alternative covariance matrix computed with vcov.tls
- **robToMiss** If TRUE, it computes the robust to misspecification covariance matrix
- **...** Other arguments when summary is applied to another class object

Value

It returns a list with the parameter estimates and their standard deviations, t-stat and p-values. It also returns the J-test and p-value for the null hypothesis that $E(g(\theta, X)) = 0$

References


Examples

```r
# GMM #
set.seed(444)
n = 500
```
sysGmm

Generalized method of moment estimation for system of equations

Description

Functions to estimate a system of equations based on GMM.

Usage

sysGmm(g, h, wmatrix = c("optimal","ident"),
vcov=c("MDS", "HAC", "CondHom", "TrueFixed"),
kernel=c("Quadratic Spectral","Truncated","Bartlett","Parzen","Tukey-Hanning"),
crit=1e-7,bw = bwAndrews, prewhite = FALSE, ar.method = "ols", approx="AR(1)",
tol = 1e-7, model=TRUE, X=FALSE, Y=FALSE, centeredVcov = TRUE,
weightsMatrix = NULL, data, crossEquConst = NULL, commonCoef = FALSE)
five(g, h, commonCoef = FALSE, data = NULL)
threeSLS(g, h, commonCoef = FALSE, data = NULL)
sur(g, commonCoef = FALSE, data = NULL)
randEffect(g, data = NULL)
Arguments

- **g**: A possibly named list of formulas
- **h**: A formula if the same instruments are used in each equation or a list of formulas.
- **wmatrix**: Which weighting matrix should be used in the objective function. By default, it is the inverse of the covariance matrix of \( g(\theta, x) \). The other choice is the identity matrix.
- **vcov**: Assumption on the properties of the moment vector. By default, it is a martingale difference sequence. "HAC" is for weakly dependent processes and "Cond-Hom" implies conditional homoscedasticity. The option "TrueFixed" is used only when the matrix of weights is provided and it is the optimal one.
- **kernel**: type of kernel used to compute the covariance matrix of the vector of sample moment conditions (see kernHAC for more details)
- **crit**: The stopping rule for the iterative GMM. It can be reduce to increase the precision.
- **bw**: The method to compute the bandwidth parameter. By default it is bwAndrews which is proposed by Andrews (1991). The alternative is bwNeweyWest of Newey-West(1994).
- **prewhite**: logical or integer. Should the estimating functions be prewhitened? If TRUE or greater than 0 a VAR model of order as.integer(prewhite) is fitted via ar with method "ols" and demean = FALSE.
- **ar.method**: character. The method argument passed to ar for prewhitening.
- **approx**: A character specifying the approximation method if the bandwidth has to be chosen by bwAndrews.
- **tol**: Weights that exceed tol are used for computing the covariance matrix, all other weights are treated as 0.
- **model, X, Y**: logical. If TRUE the corresponding components of the fit (the model frame, the model matrix, the response) are returned if g is a formula.
- **centeredVcov**: Should the moment function be centered when computing its covariance matrix. Doing so may improve inference.
- **weightsMatrix**: It allows users to provide gmm with a fixed weighting matrix. This matrix must be \( q \times q \), symmetric and strictly positive definite. When provided, the type option becomes irrelevant.
- **data**: A data.frame or a matrix with column names (Optional).
- **commonCoef**: If true, coefficients across equations are the same
- **crossEquConst**: Only used if the number of regressors are the same in each equation. It is a vector which indicates which coefficient are constant across equations. The order is 1 for Intercept and 2 to k as it is formulated in the formulas g. Setting it to 1:k is equivalent to setting commonCoef to TRUE.

Details

This set of functions implement the estimation of system of equations as presented in Hayashi (2000)
Value

'sysGmm' returns an object of 'class' "sysGmm"
The functions 'summary' is used to obtain and print a summary of the results. It also compute the J-test of overidentifying restriction

The object of class "sysGmm" is a list containing at least:

- coefficients: list of vectors of coefficients for each equation
- residuals: list of the residuals for each equation.
- fitted.values: list of the fitted values for each equation.
- vcov: the covariance matrix of the stacked coefficients
- objective: the value of the objective function $\| \text{var}(\hat{g})^{-1/2} \hat{g} \|^2$
- terms: The list of terms objects for each equation
- call: the matched call.
- y: If requested, a list of response variables.
- x: if requested, a list of the model matrices.
- model: if requested (the default), a list of the model frames.

References


Examples

data(wage)

eq1 <- LW~S+IQ+EXPR
eq2 <- LW0~S0+IQ+EXPR0

res2 <- sysGmm(eq2, data=wage)
symanly(res22)
Description

Function to estimate a linear model by the two stage least squares method.

Usage

tsls(g, x, data)

Arguments

g A formula describing the linear regression model (see details below).
x The matrix of instruments (see details below).
data A data.frame or a matrix with column names (Optional).

Details

The function just calls \texttt{gmm} with the option \texttt{vcov="iid"}. It just simplifies the implementation of 2SLS. The users don’t have to worry about all the options offered in \texttt{gmm}. The model is

\[ Y_i = X_i \beta + u_i \]

In the first step, \texttt{lm} is used to regress $X_i$ on the set of instruments $Z_i$. The second step also uses \texttt{lm} to regress $Y_i$ on the fitted values of the first step.

Value

'tsls' returns an object of 'class' ""tsls" which inherits from class ""gmm"".
The functions 'summary' is used to obtain and print a summary of the results. It also compute the J-test of overidentifying restriction

The object of class "gmm" is a list containing at least:
coefficients  $k \times 1$ vector of coefficients
residuals  the residuals, that is response minus fitted values if "g" is a formula.
fitted.values  the fitted mean values if "g" is a formula.
vcov  the covariance matrix of the coefficients
objective  the value of the objective function $\|\text{var}(\hat{g})^{-1/2}\hat{g}\|^2$
terms  the terms object used when g is a formula.
call  the matched call.
y  if requested, the response used (if "g" is a formula).
x  if requested, the model matrix used if "g" is a formula or the data if "g" is a function.
model  if requested (the default), the model frame used if "g" is a formula.
algoInfo  Information produced by either optim or nlminb related to the convergence if "g" is a function. It is printed by the summary.gmm method.

References

Examples

```r
n <- 1000
e <- arima.sim(n,model=list(ma=.9))
C <- runif(n,0,5)
Y <- rep(0,n)
for (i in 2:n){
  Y[i] = 1 + 2*C[i] + 0.9*Y[i-1] + e[i]
}
Yt <- Y[5:n]
X <- cbind(C[5:n],Y[4:(n-1)])
Z <- cbind(C[5:n],Y[3:(n-2)],Y[2:(n-3)],Y[1:(n-4)])
res <- tsls(Yt~X,-Z)
res
```

vcov  
Variance-covariance matrix of GMM or GEL

Description

It extracts the matrix of variances and covariances from gmm or gel objects.
Usage

## S3 method for class 'gmm'
vcov(object, ...)

## S3 method for class 'gel'
vcov(object, lambda = FALSE, ...)

## S3 method for class 'tsls'
vcov(object, type = c("Classical", "HC0", "HC1", "HAC"),
      hacProp = list(), ...)

## S3 method for class 'ategel'
vcov(object, lambda = FALSE, robToMiss = TRUE, ...)

Arguments

- **object**: An object of class \texttt{gmm} or \texttt{gel} returned by the function \texttt{gmm} or \texttt{gel}
- **lambda**: If set to TRUE, the covariance matrix of the Lagrange multipliers is produced.
- **type**: Type of covariance matrix for the meat
- **hacProp**: A list of arguments to pass to \texttt{kernHAC}
- **robToMiss**: If TRUE, it computes the robust to misspecification covariance matrix
- **...**: Other arguments when \texttt{vcov} is applied to another class object

Details

For \texttt{tsls()}, if \texttt{vcov} is set to a different value than "Classical", a sandwich covariance matrix is computed.

Value

A matrix of variances and covariances

Examples

```r
# GMM #
n = 500
phi <- c(.2, .7)
theta <- 0
sd <- .2
x <- matrix(arima.sim(n = n, list(order = c(2,0,1), ar = phi, ma = theta, sd = sd)), ncol = 1)
y <- x[7:n]
ym1 <- x[6:(n-1)]
ym2 <- x[5:(n-2)]
H <- cbind(x[4:(n-3)], x[3:(n-4)], x[2:(n-5)], x[1:(n-6)])
g <- y ~ ym1 + ym2
x <- H
res <- gmm(g, x)
vcov(res)
```
## Wage Data

### Description

Data used to measure return to education by Griliches (1976)

### Usage

data(wage)

### Format

A data frame containing 20 cross-sectional vectors.

- **AGE, AGE80**  Age in 1969 and 1980 respectively
- **EXPR, EXPR80** Working experience in 1969 and 1980 respectively
- **IQ**  IQ measure of the individual
- **KWW**  A test score
- **LW, LW80**  Log wage in 1969 and 1980 respectively
- **MED**  Mother education
- **MRT, MRT80**
- **RNS, RNS80**
- **S, S80**  Schooling in 1969 and 1980 respectively
- **SMSA, SMSA80**
- **TENURE, TENURE80**  Tenure in 1969 and 1980 respectively
- **YEAR**

### Source

http://fhayashi.fc2web.com/datasets.htm
Index

*Topic datasets
  Finance, 14
  Growth, 34
  nsw, 38
  wage, 54

  ar, 7, 19, 28, 44, 49
  ATEgel, 2, 36

  bread, 5
  bwAndrews, 7, 19, 49
  bwNeweyWest, 19, 49
  bwWilhelm, 6

  charStable, 8
  checkConv (ATEgel), 2
  coef, 9
  confint, 10
  constrOptim, 3, 19–21, 24, 25, 28

  estfun, 12
  evalGel, 20
  evalGel (gel), 18
  evalGmm (gmm), 26

  Finance, 14
  FinRes, 15
  fitted, 15
  five (sysGmm), 48
  formula, 17

  gel, 4, 9, 10, 15–17, 18, 20, 23, 25, 40–42, 44, 47, 53
  getDat, 23
  getLamb, 3, 19, 21, 24
  getModel, 25
  gmm, 9, 10, 12, 15–17, 20, 23, 26, 29, 35, 38, 40–42, 47, 51, 53
  gmmWithConst (gmm), 26
  Growth, 34

  kernHAC, 7, 12, 13, 19, 27, 49, 53
  KTest, 35
  lm, 20, 29, 51
  marginal, 36
  model.matrix.tsls (estfun), 12
  momentEstim, 37
  nlmInb, 3, 4, 19, 20, 24, 28, 30, 52
  nsw, 38
  optim, 3, 4, 19–21, 24, 27–30, 52
  optimize, 19–21, 28
  panel.smooth, 40
  par, 40
  plot, 39
  points, 40
  print, 41
  print.confint (confint), 10
  print.gmmTests (KTest), 35
  print.specTest (specTest), 45
  print.summary.gel (summary), 46
  print.summary.gmm (summary), 46
  print.summary.sysGmm (summary), 46
  print.summary.tsls (summary), 46
  randEffect (sysGmm), 48
  residuals, 42

  sandwich, 13
  smoothG, 43
  specTest, 45
  summary, 46
  summary.ategel (summary), 46
  summary.gel, 47
  summary.gmm, 47
  summary.sysGmm (summary), 46
  summary.sysGmm (summary), 46

55
summary.tsls(summary), 46
sur(sysGmm), 48
sysGmm, 23, 48

terms, 21, 30, 50, 52
threeSLS(sysGmm), 48
tsls, 12, 51

vcov, 52
vcov.tsls, 13
vcovHAC, 13

wage, 54
weightsAndrews, 44