Package ‘gmm’

September 7, 2009

Version 1.1-1

Date 2009-09-03

Title Generalized Method of Moments and Generalized Empirical Likelihood

Author Pierre Chausse <pierre.chausse@uqam.ca>

Maintainer Pierre Chausse <pierre.chausse@uqam.ca>

Description It is a complete suite to estimate models based on moment conditions. It includes the two step Generalized method of moments (GMM) of Hansen(1982), the iterated GMM and continuous updated estimator (CUE) of Hansen-Eaton-Yaron(1996) and several methods that belong to the Generalized Empirical Likelihood (GEL) family of estimators, as presented by Smith(1997), Kitamura(1997), Newey-Smith(2004) and Anatolyev(2005).

Depends R (>= 2.0.0)

Suggests mvtnorm, car, fBasics, MASS, timeDate, timeSeries

Imports stats

Repository CRAN

Repository/R-Forge/Project gmm

Repository/R-Forge/Revision 4

Date/Publication 2009-09-07 19:17:39

R topics documented:

charStable 2
coef.gel 3
coef.gmm 4
confint.gel 5
confint.gmm 6
Finance 7

fitted.gel 8
fitted.gmm 9
formula.gel 10
formula.gmm 11
gel 12
get_dat 16
get_lamb 17
gmm 18
HAC 22
kweights2 24
plot.gel 25
plot.gmm 26
print.gel 27
print.gmm 28
print.summary.gel 29
print.summary.gmm 30
residuals.gel 31
residuals.gmm 32
rho 32
smooth_g 33
summary.gel 35
summary.gmm 36
vcov.gel 37
vcov.gmm 38
weightsAndrews2 39

Index **41**

`charStable` *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 τ is a vector of real numbers, i is the imaginary number, x is a stable random variable with parameters θ = $(\alpha, \beta, \gamma, \delta)$ and `pm` is the type of parametrization. The vector of parameters are respectively the characteristic exponent, the skewness, the scale and the location parameters. 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

Nolan J. P. (2009), Stable Disttributions. *Math/Stat Department, American University*. URL http://academic2.american.edu/~jpnolan/stable/stable.html.

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)
```

`coef.gel` *Coefficient of GEL*

Description

It extracts the coefficients from `gel` objects.

Usage

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

Arguments

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

Value

Vector of coefficients

Examples

```
n = 500
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 <- 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)

res <- gel(g,x,t0)

coef(res)
coef(res,lambda=TRUE)
```

---

coef.gmm — *Coefficients of GMM*

Description

It extracts the coefficients from `gmm` objects.

Usage

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

Arguments

`object`	An object of class gmm returned by the function `gmm`
`...`	Other arguments when `coef` is applied to another classe object

Value

Vector of coefficients

Examples

```
n = 500
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 <- 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)
coef(res)
```

`confint.gel` *Confidence intervals for GEL*

Description

It produces confidence intervals for the coefficients and the lambdas from `gel` estimation.

Usage

```
## S3 method for class 'gel':
confint(object, parm, level=0.95, lambda=FALSE, ...)
```

Arguments

`object`	An object of class `gel` returned by the function `gel`
`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.
`...`	Other arguments when `confint` is applied to another classe object

Value

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

References

Hansen, L.P. (1982), Large Sample Properties of Generalized Method of Moments Estimators. *Econometrica*, **50**, 1029-1054,

Hansen, L.P. and Heaton, J. and Yaron, A.(1996), Finit-Sample Properties of Some Alternative GMM Estimators. *Journal of Business and Economic Statistics*, **14** 262-280.

Examples

```
n = 500
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 <- 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)

res <- gel(g,x,t0)

confint(res)
confint(res,level=0.90)
confint(res,lambda=TRUE)
```

`confint.gmm` *Confidence intervals for gmm*

Description

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

Usage

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

Arguments

`object`	An object of class `gmm` returned by the function `gmm`
`level`	The confidence level
`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.
`...`	Other arguments when `confint` is applied to an other classe object

Value

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

References

Hansen, L.P. (1982), Large Sample Properties of Generalized Method of Moments Estimators. *Econometrica*, **50**, 1029-1054,

Hansen, L.P. and Heaton, J. and Yaron, A.(1996), Finit-Sample Properties of Some Alternative GMM Estimators. *Journal of Business and Economic Statistics*, **14** 262-280.

Examples

```
n = 500
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 <- 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)

confint(res)
confint(res,level=0.90)
```

`Finance` *Returns on selected stocks*

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, compagny symboles are used.

WMK Returns of WMK

UIS Returns of UIS

ORB Returns of ORB

MAT Returns of MAT

ABAX Returns of ABAX

T Returns of T

EMR Returns of EMR

JCS Returns of JCS

VOXX Returns of VOXX

ZOOM Returns of ZOOM

TDW Returns of TDW

ROG Returns of ROG

GGG Returns of GGG

PC Returns of PC

GCO Returns of GCO

EBF Returns of EBF

F Returns of F

FNM Returns of FNM

NHP Returns of NHP

AA Returns of AA

rf Risk-free rate,

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.

Source

http://ca.finance.yahoo.com/ and http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/

`fitted.gel` *Fitted values of GEL*

Description

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

Usage

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

Arguments

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

Value

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

Examples

```
# GEL can deal with endogeneity problems

n = 200
phi<-c(.2,.7)
thet <- 0.2
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)
```

`fitted.gmm` *Fitted values of GMM*

Description

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

Usage

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

Arguments

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

Value

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

Examples

```
# 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*rnorm(n) + x

res <- gmm(y~x,x)
plot(x,y, main="Fitted model with GMM")
lines(x,fitted(res),col=2,)
legend("topright",c("Y","Yhat"),col=1:2,lty=c(1,1))
```

---

formula.gel — *Formula method for gel objects*

Description

Method to extract the formula from `gel` objects produced by `gel`.

Usage

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

Arguments

`x`	An object of class `gel` returned by the function `gel`
`...`	Other arguments to pass to other methods

Examples

```
n = 200
phi<-c(.2,.7)
thet <- 0.2
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))
formula(res)
```

`formula.gmm` *Formula method for gmm objects*

Description

Method to extract the formula from `gmm` objects produced by `gmm`.

Usage

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

Arguments

`x`	An object of class `gmm` returned by the function `gmm`
`...`	Other arguments to pass to other methods

Examples

```
# 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*rnorm(n) + x

res <- gmm(y~x,x)
formula(res)
```

gel *Generalized Empirical Likelihood estimation*

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,gradv=NULL,smooth=FALSE,type=c("EL","ET","CUE","ETEL"), vcov=c("HAC","
    kernel = c("Bartlett", "Parzen", "Truncated", "Tukey-Hanning"), bw=bwAndrews2,
    approx = c("AR(1)", "ARMA(1,1)"), prewhite = 1, ar.method = "ols", tol_weights
    tol_lam=1e-9, tol_obj = 1e-9, tol_mom = 1e-9,maxiterlam=1000,constraint=FALSE,
    optfct=c("optim","optimize","nlminb"), optlam=c("iter","numeric"), model=TRUE,
```

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 $i = 1, ...q$ and $t = 1, ..., n$. 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 θ is one, see the argument "optfct".
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).
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 θ. By default, the numerical algorithm `numericDeriv` is used. It is of course strongly suggested to provide this function when it is possible. This gradiant is used compute the asymptotic covariance matrix of $\hat{\theta}$. If "g" is a formula, the gradiant is not required (see the details below).
smooth	If set to TRUE, the moment function is smoothed as proposed by Kitamura(1997)
type	"EL" for empirical likelihood, "ET" for exponential tilting, "CUE" for continuous updated estimator and "ETEL" for exponentially tilted empirical likelihood of Schennach(2007).
vcov	Assumption on the properties of the random vector x. By default, x is a weakly dependant process. The "iid" option will only avoid using the HAC matrix to compute the covariance matrix of the parameter.
kernel	type of kernel used to compute the covariance matrix of the vector of sample moment conditions (see `HAC` for more details) and to smooth the moment conditions if "smooth" is set to TRUE.
bw	The method to compute the bandwidth parameter. By default it is `bwAndrews2` which is proposed by Andrews (1991). The alternative is `bwNeweyWest2` 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 `bwAndrews2`.
`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 λ between two iterations. The algorithm stops when $\|\lambda_i - \lambda_{i-1}\|$ reaches `tol_lamb` (see `get_lamb`)
`maxiterlam`	The algorithm to compute λ stops if there is no convergence after "maxiterlam" iterations (see `get_lamb`).
`tol_obj`	Tolerance for the gradiant of the objective function to compute λ (see `get_lamb`).
`optfct`	Only when the dimension of θ 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 1×2 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 $\sum_{t=1}^n p_t g(\theta(x_t) = 0$, where $p_t = \frac{1}{n}D\rho(<g_t, \lambda>)$ is the implied probability. It adds a penalty if the solution diverges from its goal.
`optlam`	The default is "iter" which solves for λ using the Newton iterative method `get_lamb`. If set to "numeric", the algorithm `optim` is used to compute λ instead.
`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.
`...`	More options to give to `optim`, `optimize` or `constrOptim`.

Details

`weightsAndrews2` and `bwAndrews2` are simply modified version of `weightsAndrews` and `bwAndrews` from the package sandwich. The modifications have been made so that the argument x can be a matrix instead of an object of class lm or glm. The details on how is works can be found on the sandwich manual.

If we want to estimate a model like $Y_t = \theta_1 + X_{2t}\theta_2 + ... + X_k\theta_k + \epsilon_t$ using the moment conditions $Cov(\epsilon_t H_t) = 0$, where H_t is a vector of Nh instruments, than we can define "g" like we do for `lm`. We would have `g = y~x2+x3+...+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 gradiant 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 $\hat{\theta} = \arg\min \left[\arg\max_\lambda \frac{1}{n} \sum_{t=1}^n \rho(< g(\theta, x_t), \lambda >) - \rho(0)\right]$

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 `get_lamb`)
`conv_mes`	Convergence message for "lambda" (see `get_lamb`)
`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

Anatolyev, S. (2005), GMM, GEL, Serial Correlation, and Asymptotic Bias. *Econometrica*, **73**, 983-1002.

Andrews DWK (1991), Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation. *Econometrica*, **59**, 817–858.

Kitamura, Yuichi (1997), Empirical Likelihood Methods With Weakly Dependent Processes. *The Annals of Statistics*, **25**, 2084-2102.

Newey, W.K. and Smith, R.J. (2004), Higher Order Properties of GMM and Generalized Empirical Likelihood Estimators. *Econometrica*, **72**, 219-255.

Newey WK & West KD (1987), A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, **55**, 703–708.

Newey WK & West KD (1994), Automatic Lag Selection in Covariance Matrix Estimation. *Review of Economic Studies*, **61**, 631-653.

Schennach, Susanne, M. (2007), Point Estimation with Exponentially Tilted Empirical Likelihood. *Econometrica*, **35**, 634-672.

Zeileis A (2006), Object-oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16. URL http://www.jstatsoft.org/v16/i09/.

Examples

```
# First, an exemple with the fonction g()

g <- function(tet,x)
        {
        n <- nrow(x)
        u <- (x[7:n] - tet[1] - tet[2]*x[6:(n-1)] - tet[3]*x[5:(n-2)])
        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)
        }

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)
        }
n = 200
phi<-c(.2,.7)
thet <- 0.2
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)

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

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))
summary(res)
```

get_dat	*Extracting data from a formula*

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 gmm and gel.

Usage

```
get_dat(formula,h)
```

Arguments

formula	A formula that defines the linear model to be estimated (see details).
h	A $n \times nh$ matrix of intruments(see details).

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

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

```
n = 500
phi<-c(.2,.7)
thet <- 0.2
sd <- .2
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)])

x <- get_dat(y~ym1+ym2,H)
```

get_lamb *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

```
get_lamb(g,tet,x,type=c('EL','ET','CUE'),tol_lam=1e-12,maxiterlam=1000,tol_obj=1e-7
```

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 $i = 1, ...q$ and $t = 1, ..., n$. This matrix is then used to build the q sample moment conditions.
`tet`	A $k \times 1$ vector of parameters at which the function $g(\theta, x)$ has to be evaluated
`x`	The matrix or vector of data from which the function $g(\theta, x)$ is computed.
`type`	"EL" for empirical likelihood, "ET" for exponential tilting and "CUE" for continuous updated estimator.
`tol_lam`	Tolerance for λ between two iterations. The algorithm stops when $\|\lambda_i - \lambda_{i-1}\|$ reaches `tol_lam`
`maxiterlam`	The algorithm stops if there is no convergence after "maxiterlam" iterations.
`tol_obj`	Tolerance for the gradiant of the objective function. The algorithm returns a non-convergence message if $\max(\|gradiant\|)$ does not reach `tol_obj`. It helps the `gel` algorithm to select the right space to look for θ

Details

It solves the problem $\frac{1}{n}\sum_{t=1}^{n} D\rho(< g(\theta, x_t), \lambda >)g(\theta, x_t) = 0$.

Value

lambda: A $q \times 1$ vector of Lagrange multipliers which solve the system of equations given above. singular: 0 for a normal solution, 1 if the algorithm does not converge and 2 if the algorithm produces a singular system, NaN or Inf values. `conv_mes`: A message with details about the convergence.

References

Newey, W.K. and Smith, R.J. (2004), Higher Order Properties of GMM and Generalized Empirical Likelihood Estimators. *Econometrica*, **72**, 219-255.

Examples

```
g <- function(tet,x)
        {
        n <- nrow(x)
        u <- (x[7:n] - tet[1] - tet[2]*x[6:(n-1)] - tet[3]*x[5:(n-2)])
        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)
thet <- 0.2
sd <- .2
x <- matrix(arima.sim(n=n,list(order=c(2,0,1),ar=phi,ma=thet,sd=sd)),ncol=1)
get_lamb(g,c(0,phi),x,type="EL")
```

---

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

```
gmm(g,x,t0=NULL,gradv=NULL, type=c("twoStep","cue","iterative"),
wmatrix = c("optimal","ident"),  vcov=c("HAC","iid"),
kernel=c("Quadratic Spectral","Truncated", "Bartlett",
"Parzen", "Tukey-Hanning"),crit=10e-7,bw = bwAndrews2,
prewhite = FALSE, ar.method = "ols", approx="AR(1)",tol = 1e-7,
itermax=100,optfct=c("optim","optimize","nlminb"), model=TRUE, X=FALSE, Y=FALSE, ..
```

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 $i = 1, ...q$ and $t = 1, ..., n$. 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 (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 θ is one, see the argument "optfct".

`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 θ. By default, the numerical algorithm `numericDeriv` is used. It is of course strongly suggested to provide this function when it is possible. This gradiant is used compute the asymptotic covariance matrix of $\hat{\theta}$. If "g" is a formula, the gradiant 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 θ
`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.
`kernel`	type of kernel used to compute the covariance matrix of the vector of sample moment conditions (see `HAC` for more details)
`crit`	The stoping 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 `bwAndrews2` which is proposed by Andrews (1991). The alternative is `bwNeweyWest2` 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 `bwAndrews2`.
`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 θ 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 1×2 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=`.
`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.
`...`	More options to give to `optim`.

Details

`weightsAndrews2` and `bwAndrews2` are simply modified version of `weightsAndrews` and `bwAndrews` from the package sandwich. The modifications have been made so that the argument x can be a matrix instead of an object of class lm or glm. The details on how is works can be found on the sandwich manual.

If we want to estimate a model like $Y_t = \theta_1 + X_{2t}\theta_2 + \cdots + X_k\theta_k + \epsilon_t$ using the moment conditions $Cov(\epsilon_t H_t) = 0$, where H_t is a vector of Nh instruments, than we can define "g" like we do for `lm`.

We would have $g = y\tilde{}x2 + x3 + \cdots + xk$ 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 gradiant 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.

The following explains the last example bellow. Thanks to Dieter Rozenich, a student from the Vienna Universtiy 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 (μ, σ) 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 (μ, σ). The third moment condition comes from the third derivative of the moment generating function (MGF)

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

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):

$$1 \qquad 0$$

$$-2\mu + 2x \qquad 2\sigma$$

$$-3\mu^2 - 3\sigma^2 \qquad -6\mu\sigma$$

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 overidentying 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 $\|var(\bar{g})^{-1/2}\bar{g}\|^2$

`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.

References

Zeileis A (2006), Object-oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16. URL http://www.jstatsoft.org/v16/i09/.

Andrews DWK (1991), Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation. *Econometrica*, **59**, 817–858.

Newey WK & West KD (1987), A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, **55**, 703–708.

Newey WK & West KD (1994), Automatic Lag Selection in Covariance Matrix Estimation. *Review of Economic Studies*, **61**, 631-653.

Hansen, L.P. (1982), Large Sample Properties of Generalized Method of Moments Estimators. *Econometrica*, **50**, 1029-1054,

Hansen, L.P. and Heaton, J. and Yaron, A.(1996), Finit-Sample Properties of Some Alternative GMM Estimators. *Journal of Business and Economic Statistics*, **14** 262-280.

Examples

```
## 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 linear.hypothesis from the car package
## The CAPM can be tested as follows:

library(car)
linear.hypothesis(res,cbind(diag(10),matrix(0,10,10)),rep(0,10))

# The CAPM of Black
g <- function(theta,x) {
        e <- x[,2:11]-theta[1]-(x[,1]-theta[1])%*%matrix(theta[2:11],1,10)
        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"]-rf
f3 <- Finance[1:300,"smb"]-rf
h <- cbind(f1,f2,f3)
res2 <- gmm(z~f1+f2+f3,x=h)
coef(res2)
summary(res2)$coefficients

## 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]*(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*(-tet[1]+mean(x)), -3*tet[1]^2-3*tet[2]^2,0, 2*tet[2],-6*
        return(jacobian)
        }
# Now we want to estimate the two parameters using the GMM.
gmm(g,x,c(0,0),grad=Dg)
```

HAC *Covariance matrix of weakly dependent time series*

Description

Function to compute a consistent covariance matrix of the sample mean of a random vector of time series. It is consistent in presence of heteroscedasticity and autocorrelation.

Usage

```
HAC(x, weights = weightsAndrews2, bw = bwAndrews2, prewhite = FALSE, ar.method = "o
    kernel=c("Quadratic Spectral", "Truncated", "Bartlett", "Parzen", "Tukey-Hannin
    approx="AR(1)",tol = 1e-7)
```

Arguments

`x`	a $n \times q$ matrix of time series, where n is the sample size.
`weights`	The method to compute the kernel weights. For now, weightsAndrews2 is the only one possible. I leave the option there because I am planning to give more choices in futur versions of the package.
`bw`	The method to compute the bandwidth parameter. By default it is `bwAndrews2` which is proposed by Andrews (1991). The alternative is `bwNeweyWest2` 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.
`kernel`	The choice of kernel
`approx`	a character specifying the approximation method if the bandwidth has to be chosen by `bwAndrews2`.
`tol`	numeric. Weights that exceed `tol` are used for computing the covariance matrix, all other weights are treated as 0.

Details

`HAC` is simply a modified version of `meatHAC` from the package sandwich. The modifications have been made so that the argument x can be a matrix instead of an object of class lm or glm. The details on how is works can be found on the sandwich manual.

Value

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 gmm() but can also be used by itself.

References

Zeileis A (2006), Object-oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16. URL `http://www.jstatsoft.org/v16/i09/`.

Newey WK & West KD (1987), A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, **55**, 703–708.

Newey WK & West KD (1994), Automatic Lag Selection in Covariance Matrix Estimation. *Review of Economic Studies*, **61**, 631-653.

Examples

```
x <- arima.sim(n=200,list(order=c(1,0,1),ar=.5,ma=.3,sd=.5))
y <- .4*x+rnorm(200)
x <- cbind(x,y)
vcov <- HAC(x)
```

---

kweights2 *Kernel Weights*

Description

Kernel weights for kernel-based heteroskedasticity and autocorrelation consistent (HAC) covariance matrix estimators as introduced by Andrews (1991). This function comes from the `sandwich` package.

Usage

```
kweights2(x, kernel = c("Truncated", "Bartlett", "Parzen",
  "Tukey-Hanning", "Quadratic Spectral"), normalize = FALSE)
```

Arguments

`x`	numeric.
`kernel`	a character specifying the kernel used. All kernels used are described in Andrews (1991).
`normalize`	logical. If set to `TRUE` the kernels are normalized as described in Andrews (1991).

Value

Value of the kernel function at `x`.

References

Andrews DWK (1991), Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation. *Econometrica*, **59**, 817–858.

plot.gel — *Plot Diagnostics for a gel Object*

Description

It is a plot method for `gel` objects.

Usage

```
## S3 method for class 'gel':
plot(x, which = c(1L: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"))
```

Arguments

`x`	`gel` object, typically result of `gel`.
`which`	if a subset of the plots is required, specify a subset of the numbers `1:4`.
`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 *ask*ed 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

```
n = 500
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 <- 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)

res <- gel(g,x,t0)

plot(res,which=3)
plot(res,which=4)
```

`plot.gmm` *Plot Diagnostics for a gmm Object*

Description

It is a plot method for `gmm` objects.

Usage

```
## S3 method for class 'gmm':
plot(x, which = c(1L: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`	`gmm` object, typically result of `gmm`.
`which`	if a subset of the plots is required, specify a subset of the numbers `1:3`.
`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 *ask*ed 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 `gmm` 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` and the third is the dependant variable y with its mean $\hat{y}$ (the fitted values).

Examples

```
n = 500
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 <- 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)

plot(res,which=3)
```

`print.gel` *Printing a gel object*

Description

It is a printing method for `gel` objects.

Usage

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

Arguments

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

Value

It prints some results from the estimation like the coefficients.

Examples

```
n = 500
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 <- 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)

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

---

print.gmm *Printing a gmm object*

Description

It is a printing method for gmm objects.

Usage

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

Arguments

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

Value

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

Examples

```
n = 500
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 <- 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)
print(res)
```

`print.summary.gel` *Printing the summary of gel*

Description

It is a printing method for `summary.gel` objects.

Usage

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

Arguments

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

Value

It prints some results from `summary.gel`

Examples

```
n = 500
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 <- 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)

res <- gel(g,x,t0)
summary(res)
```

`print.summary.gmm` *Printing the summary of gmm*

Description

It is a printing method for `summary.gmm` objects.

Usage

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

Arguments

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

Value

It prints some results from `summary.gmm`

Examples

```
n = 500
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 <- 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)

print(summary(res))
```

residuals.gel — *Residuals of GEL*

Description

Method to extract the residuals of the model estimated by `gel`.

Usage

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

Arguments

`object`	An object of class `gel` returned by the function `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~x` as it is done by `residuals.lm`.

Examples

```
# GEL can deal with endogeneity problems

n = 200
phi<-c(.2,.7)
thet <- 0.2
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))
e <- residuals(res)
plot(e,type='l',main="Residuals from an ARMA fit using GEL")
```

residuals.gmm *Residuals of GMM*

Description

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

Usage

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

Arguments

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

Value

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

Examples

```
# 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*rnorm(n) + x

res <- gmm(y~x,x)
plot(x,residuals(res), main="Residuals of an estimated model with GMM")
```

rho *Objective function of Generalized Empirical Likelihood (GEL)*

Description

It computes the objective function of GEL, its first and second analytical derivatives. It is called by gel.

Usage

```
rho(x,lamb,derive=0,type=c("EL","ET","CUE"),drop=TRUE)
```

Arguments

x	A $n \times q$ matrix with typical element (t, i), $g_i(\theta, x_t)$
lamb	A $q \times 1$ vector of lagrange multipliers
derive	0 for the objective function, 1 for the first derivative with respect to λ and 2 for the second derivative with respect to λ.
type	"EL" for empirical likelihood, "ET" for exponential tilting and "CUE" for continuous updated estimator.
drop	Because the solution may not be in the domain of $\rho(v)$ $\forall t$ in small sample, we can drop those observations to avoid the return of NaN

Details

The objective function is $\frac{1}{n}\sum_{t=1}^{n}\rho(< g(\theta, x_t), \lambda >)$, where $\rho(v) = \log(1-v)$ for empirical likelihood, $-e^v$ for exponential tilting and $(-v - 0.5v^2)$ for continuous updated estimator.

Value

'rho' returns a scalar if "derive=0", a q vector if "derive" = 1 and a $q \times q$ matrix if derive = 2.

References

Newey, W.K. and Smith, R.J. (2004), Higher Order Properties of GMM and Generalized Empirical Likelihood Estimators. *Econometrica*, **72**, 219-255.

Hansen, L.P. and Heaton, J. and Yaron, A.(1996), Finit-Sample Properties of Some Alternative GMM Estimators. *Journal of Business and Economic Statistics*, **14** 262-280.

---

smooth_g *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

```
smooth_g(x, bw = bwAndrews2, 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=bwNeweyWest2 (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 `weightsAndrews2` 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 `bwAndrews2`.
- `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

`HAC` is simply a modified version of `meatHAC` from the package sandwich. The modifications have been made so that the argument x can be a matrix instead of an object of class lm or glm. The details on how is works can be found on the sandwich manual.

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.

If the vector of weights is provided, it gives only one side weights. For exemple, 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

Anatolyev, S. (2005), GMM, GEL, Serial Correlation, and Asymptotic Bias. *Econometrica*, **73**, 983-1002.

Andrews DWK (1991), Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation. *Econometrica*, **59**, 817–858.

Kitamura, Yuichi (1997), Empirical Likelihood Methods With Weakly Dependent Processes. *The Annals of Statistics*, **25**, 2084-2102.

Zeileis A (2006), Object-oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16. URL `http://www.jstatsoft.org/v16/i09/`.

Examples

```
g <- function(tet,x)
        {
        n <- nrow(x)
        u <- (x[7:n] - tet[1] - tet[2]*x[6:(n-1)] - tet[3]*x[5:(n-2)])
        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)
thet <- 0.2
sd <- .2
x <- matrix(arima.sim(n=n,list(order=c(2,0,1),ar=phi,ma=thet,sd=sd)),ncol=1)
gt <- g(c(0,phi),x)
sgt <- smooth_g(gt)$smoothx
plot(gt[,1])
lines(sgt[,1])
```

---

summary.gel *Method for object of class gel*

Description

It presents the results from the `gel` estimation in the same fashion as `summary` does for the `lm` class objects. It also compute the J-test, LM and LR tests for overidentifying restriction.

Usage

```
## S3 method for class 'gel':
summary(object, ...)
```

Arguments

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

Value

It returns a list with the parameter estimates and their standard deviations, t-stats and p-values. It also returns the three tests (J, LM and LR) and p-values for the null hypothesis that $E(g(\theta, X) = 0$ and several convergence codes.

References

Anatolyev, S. (2005), GMM, GEL, Serial Correlation, and Asymptotic Bias. *Econometrica*, **73**, 983-1002.

Kitamura, Yuichi (1997), Empirical Likelihood Methods With Weakly Dependent Processes. *The Annals of Statistics*, **25**, 2084-2102.

Newey, W.K. and Smith, R.J. (2004), Higher Order Properties of GMM and Generalized Empirical Likelihood Estimators. *Econometrica*, **72**, 219-255.

Examples

```
n = 500
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 <- 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)

res <- gel(g,x,t0)
summary(res)
```

---

summary.gmm — *Method for object of class gmm*

Description

It presents the results from the `gmm` estimation in the same fashion as `summary` does for the `lm` class objects for example. It also compute the J-test for overidentifying restrictions.

Usage

```
## S3 method for class 'gmm':
summary(object, ...)
```

Arguments

`object`	An object of class `gmm` returned by the function `gmm`
`...`	Other arguments when summary is applied to another classe 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

Hansen, L.P. (1982), Large Sample Properties of Generalized Method of Moments Estimators. *Econometrica*, **50**, 1029-1054,

Hansen, L.P. and Heaton, J. and Yaron, A.(1996), Finit-Sample Properties of Some Alternative GMM Estimators. *Journal of Business and Economic Statistics*, **14** 262-280.

Examples

```
n = 500
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 <- 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)

summary(res)
```

---

vcov.gel — *Variance-covariance matrix of GEL*

Description

It extracts the matrix of variances and covariances from `gel` objects.

Usage

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

Arguments

`object`	An object of class `gel` returned by the function `gel`
`lambda`	If set to TRUE, the covariance matrix of the Lagrange multipliers is produced.
`...`	Other arguments when `vcov` is applied to an other classe object

Value

A matrix of variances and covariances

Examples

```
n = 500
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 <- 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)

res <- gel(g,x,t0)
vcov(res)
vcov(res,lambda=TRUE)
```

---

vcov.gmm *Variance-covariance matrix of GMM*

Description

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

Usage

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

Arguments

object	An object of class gmm returned by the function gmm
...	Other arguments when vcov is applied to another classe object

Value

A matrix of variances and covariances

Examples

```
n = 500
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 <- 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)
```

`weightsAndrews2` *Kernel weights*

Description

Function to compute the kernel weights used to compute the HAC covariance matrix

Usage

```
weightsAndrews2(x, bw = bwAndrews2, kernel = c("Quadratic Spectral",
                "Truncated", "Bartlett", "Parzen", "Tukey-Hanning"), approx = c("AR
                "ARMA(1,1)"), prewhite = 1, ar.method = "ols", tol = 1e-7, verbose

bwAndrews2(x, kernel = c("Quadratic Spectral",
           "Truncated", "Bartlett", "Parzen", "Tukey-Hanning"), approx = c("AR(1)",
           "ARMA(1,1)"), prewhite = 1, ar.method = "ols")

bwNeweyWest2(x, kernel = c("Bartlett", "Parzen",
             "Quadratic Spectral", "Truncated", "Tukey-Hanning"),
              prewhite = 1, ar.method = "ols",...)
```

Arguments

`x`	A $n \times q$ matrix of time series from which we want to compute the covariance matrix.
`bw`	The method to compute the bandwidth parameter. For now, bwAndrews2 is the only one possible. I leave the option there because I am planning to give more choices in futur versions of the package.
`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.
`verbose`	logical. Should the bandwidth parameter used be printed?
`approx`	a character specifying the approximation method if the bandwidth has to be chosen by `bwAndrews2`.
`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
`...`	It just allows to call either `bwAndrews2` or `bwNeweyWest` without having unusued arguments.

Details

`weightsAndrews2`, `bwAndrews2` and `bwNeweyWest2` are simply modified version of `weightsAndrews`, `bwAndrews` and `bwNeweyWest` from the package sandwich. The modifications have been made so that the argument x can be a matrix instead of an object of class lm or glm. The details on how is works can be found on the sandwich manual. `kweights` is the same as the one included in the package sandwich.

Value

`weightsAndrews` returns a vector of weights.

`bwAndrews` returns the selected bandwidth parameter.

References

Zeileis A (2006), Object-oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16. URL http://www.jstatsoft.org/v16/i09/.

Andrews DWK (1991), Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation. *Econometrica*, **59**, 817–858.

Newey WK & West KD (1987), A Simple, Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix. *Econometrica*, **55**, 703–708.

Newey WK & West KD (1994), Automatic Lag Selection in Covariance Matrix Estimation. *Review of Economic Studies*, **61**, 631-653.

Examples

```
set.seed(345)
x <- arima.sim(n=200,list(ordre=c(1,0,1),ar=.9,ma=.4))
y <- 2*x + rnorm(200)
x = cbind(x,y)
w <- weightsAndrews2(x, bw = bwAndrews2, kernel = "Quadratic")
plot(w,type='l')
w2 <- weightsAndrews2(x, bw = bwNeweyWest2, kernel = "Bartlett")
plot(w2,type='l')
```

Index

∗Topic **datasets**
  Finance, 7
∗Topic **regression**
  kweights2, 24
∗Topic **ts**
  kweights2, 24

ar, *12*, *19*, *23*, *34*, *39*

bwAndrews2, *12, 13*, *19*, *23*
bwAndrews2 *(weightsAndrews2)*, 39
bwNeweyWest2, *12*, *19*, *23*
bwNeweyWest2 *(weightsAndrews2)*, 39

charStable, 2
coef.gel, 3
coef.gmm, 4
confint.gel, 5
confint.gmm, 6
constrOptim, *13, 14*

Finance, 7
fitted.gel, 8
fitted.gmm, 9
formula.gel, 10
formula.gmm, 11

gel, *3*, *5*, *8*, *10*, 11, *13*, *15*, *25*, *27*, *31, 32*, *34, 35*, *37*
get_dat, 15
get_lamb, *12–14*, 16
gmm, *4*, *6*, *9*, *11*, *15*, 18, *19*, *26*, *28*, *32*, *36*, *38*

HAC, *12*, *18*, 22

kweights *(weightsAndrews2)*, 39
kweights2, 24

lm, *13*, *19*

nlminb, *13*, *19*

optim, *13, 14*, *19*
optimize, *13, 14*, *19*

panel.smooth, *25, 26*
par, *25, 26*
plot.gel, 24
plot.gmm, 25
points, *25, 26*
print.gel, 27
print.gmm, 28
print.summary.gel, 29
print.summary.gmm, 30

residuals.gel, 31
residuals.gmm, 32
rho, 32

smooth_g, 33
summary.gel, *29*, 35
summary.gmm, *30*, 36

terms, *14*, *20*

vcov.gel, 37
vcov.gmm, 38

weightsAndrews2, *13*, *34*, 39